An Efficient Background Term for 3D Reconstruction and Tracking with Smooth Surface Models

Mariano Jaimez[1,2*] Thomas J. Cashman[3] Andrew Fitzgibbon[3]
Javier Gonzalez-Jimenez[1] Daniel Cremers[2]

[1]University of Malaga [2]Technical University of Munich [3]Microsoft

Abstract

We present a novel strategy to shrink and constrain a 3D model, represented as a smooth spline-like surface, within the visual hull of an object observed from one or multiple views. This new 'background' or 'silhouette' term combines the efficiency of previous approaches based on an image-plane distance transform with the accuracy of formulations based on raycasting or ray potentials. The overall formulation is solved by alternating an inner nonlinear minimization (raycasting) with a joint optimization of the surface geometry, the camera poses and the data correspondences. Experiments on 3D reconstruction and object tracking show that the new formulation corrects several deficiencies of existing approaches, for instance when modelling non-convex shapes. Moreover, our proposal is more robust against defects in the object segmentation and inherently handles the presence of uncertainty in the measurements (e.g. null depth values in images provided by RGB-D cameras).

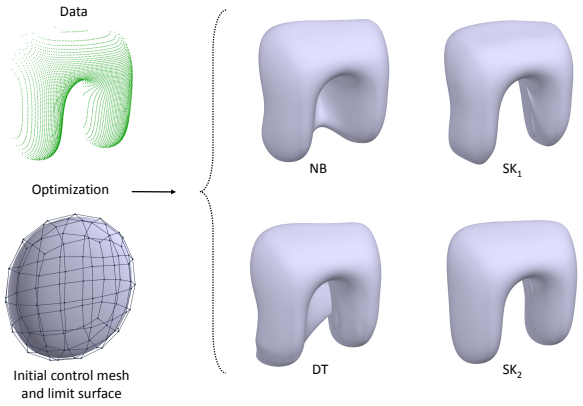


Figure 1. Modelling an arch from a single depth image and a challenging initial control mesh topology. Even with depth data, the model spills over into the background region (NB). We show how existing background terms based on the distance transform (DT) fail to capture the concavity, while our new methods based on shrinking kernels (SK_1 and SK_2) succeed.

1. Introduction

An important problem in computer vision is the recovery of 3D models from one or more images, whether RGB or depth. Examples include human body tracking [6], single-view reconstruction [13], or the acquisition of deformable object class models [3]. A dominant paradigm is to express the problem as energy minimization: find the 3D model parameters (including model shape, camera positions, etc.) which best explain the given data.

Formulations of such problems as energy minimization typically involve two key data terms: a term encouraging 'foreground' measurements within the object to be explained by the model, and a 'background' or 'empty space' term, requiring that the model does not project in front of data samples known to be outside the object. Typically the foreground terms are easily written as variants of a closest-point or closest-intensity objective, which are readily optimized using ICP [1] or lifting algorithms [7, 3]. In contrast, the background terms involve an expensive raycasting or rendering operation, or are approximated by projecting a finite subset of points on the model surface into a distance transform.

This paper's contribution is to illustrate the failings of distance-transform-based background terms, and to introduce a new formulation with the accuracy of raycasting but which admits efficient optimization using smooth-function optimizers such as Levenberg-Marquardt. The key innovation is to write raycasting as an optimization problem in its own right, and to solve the min-of-max optimization that results from combining raycasting with the foreground term.

We will demonstrate the advantages of such a formulation in two very common scenarios in computer vision: 3D reconstruction and non-rigid tracking. In both cases the ob-

*Research funded by the Spanish Government and the European Regional Development Fund (ERDF) under contract DPI2014-55826-R, the Spanish grant program FPI-MICINN 2012 and the ERC Consolidator Grant *3DReloaded*.

ject to be reconstructed or tracked will be modeled with a subdivision surface, and the input data will consist of one or multiple depth images with unknown camera positions.

2. Related Work

Given the considerable body of related literature, we focus on only a few key examples of the existing approaches.

One class of methods is volumetric or **voxel-based**. For example, the single-view reconstruction work of Töppe *et al*. [20], which imposes the hard constraint that object voxels must project into foreground regions, and expresses the reconstruction problem as energy minimization with one parameter per voxel. Recent work [15, 5] allows significant improvements in optimization, but the large state space limits its model resolution, and as noted by Oswald *et al*. [13], the absence of a thresholding theorem means that the relaxation method employed for solution may not yield the optimal Boolean labelling. KinectFusion [12] avoids an optimization over the entire volume by estimating camera position using robust ICP, followed by deterministic carving of a 3D signed distance function, but copes poorly with missing data.

This paper focuses on **mesh-based** methods, such as used by Prasad *et al*. [14] for single-view reconstruction, or Vicente and Agapito [21] in deforming a template 3D mesh to match a given image silhouette. The latter paper used a distance transform penalty for the background term, as did Ganapathi *et al*. [6] in solving the problem of human body tracking. As shown below, the distance transform term has several limitations.

We consider a smooth surface representation based on subdivision surfaces. This spline-like representation has recently been used for 3D morphable model construction [3], hand shape estimation [19] and hand tracking [18]. Subdivision surfaces have also been used to fit 3D point clouds or regular meshes [8, 11, 4, 9, 10], but our guiding example problems differ in that either the camera positions, or the mesh/object topology, or both, are unknown. Moreover, in real applications a significant percentage of the object to reconstruct or track might be missing due to lack of sufficient views and errors in the measurement process (null pixels in depth images provided by RGB-D cameras). Under these circumstances, the use of an effective and efficient background term becomes crucial.

3. Definitions and Notation

To illustrate the advantages of our proposal, we address two distinct problems: (A) generating a 3D reconstruction of an object from multiple views and (B) tracking a non-rigid object from an image sequence. In both cases, our input data comprise a set of N depth images $\{Z_i\}_{i=1}^{N}$ of a target object. A depth image is a collection of 3D points $Z_i = [\boldsymbol{p}_{ij}]_{j=1}^{M}$ associated with 2D pixel coordinates $\boldsymbol{x}_{ij}$ through the projection function $\boldsymbol{\pi} : \mathbb{R}^3 \mapsto \mathbb{R}^2$, i.e. $\boldsymbol{x}_{ij} = \boldsymbol{\pi}(\boldsymbol{p}_{ij})$. For each data point $\boldsymbol{p}_{ij}$ we also estimate a unit normal $\boldsymbol{n}_{ij}$ of the target object with $\|\boldsymbol{n}_{ij}\| = 1$. Each image i has an unknown camera pose parametrized using the twist $\boldsymbol{\xi}_i \in \mathbb{R}^6$, inducing a rotation matrix $R(\boldsymbol{\xi}_i) \in \mathrm{SO}(3) \subset \mathbb{R}^{3\times3}$ and a translation vector $\boldsymbol{t}(\boldsymbol{\xi}_i) \in \mathbb{R}^3$. The function $\boldsymbol{\pi}$ is also overloaded to project world-coordinate points $\boldsymbol{s}$ by passing the pose of the camera to which points are projected, so $\boldsymbol{\pi}(\boldsymbol{s}, \boldsymbol{\xi}_i) = \boldsymbol{\pi}\big(R(\boldsymbol{\xi}_i)\boldsymbol{s} + \boldsymbol{t}(\boldsymbol{\xi}_i)\big)$.

The pixel indices are segmented into three disjoint regions D_i, B_i and C_i to specify whether each pixel observes the target object, the background, or provides no valid depth respectively. For example, $j \in D_i$ means that 3D point $\boldsymbol{p}_{ij}$ is a foreground measurement, and $j \in C_i$ means that pixel $\boldsymbol{x}_{ij}$ has an invalid 3D point. As we shall never access $\boldsymbol{p}_{ij}$ for invalid pixels, we need not define it in this case.

To model the object we use a Catmull-Clark subdivision surface, the shape of which is defined by a control mesh comprising P control vertices $\{\boldsymbol{X}_p\}_{p=1}^{P} \subset \mathbb{R}^3$ that are referenced as the corners of F quadrilateral faces $\{\boldsymbol{Q}_f\}_{f=1}^{F} \subset \{1..P\}^4$. The subdivision surface is a mapping $\boldsymbol{s} : \Omega \mapsto \mathbb{R}^3$, where the parametric domain Ω of the surface is the union $\square \times \{1..F\}$ of F copies of the unit square $\square := [\,0,1\,] \times [\,0,1\,] \subset \mathbb{R}^2$: one copy for each face in the control mesh. The topology of the surface, and hence Ω, is held fixed throughout the optimization, and so the shape of the surface is determined purely by the control vertices $X = \{\boldsymbol{X}_p\}_{p=1}^{P}$. We therefore write the surface as $\boldsymbol{s}(\boldsymbol{u}|X)$ where $\boldsymbol{u}$ is the tuple $\boldsymbol{u} = (u_x, u_y, f) \in \Omega$. The normalized surface normal is written similarly as $\boldsymbol{s}^{\perp}(\boldsymbol{u}|X)$.

4. Optimization Problem

We present a unified framework to address either the 3D reconstruction or the non-rigid tracking of an object. It is based on two main constraints:

- The model must fit the geometric data ($\boldsymbol{p}_{ij}$ and $\boldsymbol{n}_{ij}$) computed from the depth images $\{Z_i\}$ for those pixels $j \in D_i$ where the object is *present*.
- The model should not be observable from pixels $\boldsymbol{x}_{ij}$ which are known to observe the background (i.e. $j \in B_i$), since in these locations we know the target object to be *absent*.

The remaining pixels, referenced by C_i, should not place any restriction on the model since their true depth is unknown, and we therefore have no evidence for either presence or absence of the target object.

4.1. Data Term

The first energy term measures the error in position and orientation between pixel $j \in D_i$ and a to-be-estimated

corresponding model point $\boldsymbol{u} \in \Omega$:

$$E^p_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}) = \left\| \boldsymbol{p}_{ij} - R(\boldsymbol{\xi}_i)\boldsymbol{s}(\boldsymbol{u}|X) - \boldsymbol{t}(\boldsymbol{\xi}_i) \right\|^2_T, \quad (1)$$

$$E^n_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}) = \left\| \boldsymbol{n}_{ij} - R(\boldsymbol{\xi}_i)\boldsymbol{s}^{\perp}(\boldsymbol{u}|X) \right\|^2_T, \quad (2)$$

where $\|\bullet\|_T$ represents a truncated Euclidean norm. We combine (1) and (2) into a *data term* defined as the weighted combination

$$\hat{E}^d(X, \xi) = \sum_{i=1}^{N} \sum_{j \in D_i} \min_{\boldsymbol{u}} \left(\lambda_p E^p_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}) + \lambda_n E^n_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}) \right). \quad (3)$$

This energy allows us to fit the model to the data by *lifting* [7, 3, 19] the latent model-data correspondences $\mathcal{U} = \{\boldsymbol{u}^d_{ij}\}$ for each $i = 1 \dots N$ and $j \in D_i$ to give the energy

$$E^d(X, \xi, \mathcal{U}) = \sum_{i=1}^{N} \sum_{j \in D_i} \lambda_p E^p_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}^d_{ij}) + \lambda_n E^n_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}^d_{ij}) \quad (4)$$

with $\hat{E}^d(X, \xi) \leq E^d(X, \xi, \mathcal{U})$ for all $\mathcal{U}$. We can therefore minimize (3) and fit the observed data by finding

$$\arg\min_{X, \xi, \mathcal{U}} \left\{ E^d(X, \xi, \mathcal{U}) \right\}. \quad (5)$$

4.2. Background Term

Unfortunately, solving (5) often gives poor results, because there is nothing to penalize the model spilling over the observed object silhouette (see Fig. 1 panel 'NB' or the teddy bear's legs in Fig. 4). It is therefore necessary to define an additional energy term that forces the model to remain within the visual hull of the object, as observed by the depth images.

Our goal is to have a background term that penalizes instances of the model that project into pixels $j \in B_i$ where the object is known to be absent. Essentially this is a sum of terms of the form "if any point anywhere on the model projects to pixel $\boldsymbol{x}_{ij}$, pay a penalty":

$$\sum_{i=1}^{N} \sum_{j \in B_i} \begin{cases} 1 & \text{if } \exists \boldsymbol{u} \in \Omega \text{ with } \boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}|X), \boldsymbol{\xi}_i) = \boldsymbol{x}_{ij} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

which can be re-cast as a minimization

$$\sum_{i=1}^{N} \sum_{j \in B_i} \begin{cases} 1 & \text{if } \min_{\boldsymbol{u} \in \Omega} \|\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}|X), \boldsymbol{\xi}_i) - \boldsymbol{x}_{ij}\| = 0 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

and then written in terms of the finite delta function $\perp(r) = [\![r = 0]\!]$ (in code $\perp(r) = \{\text{if } r = 0 \text{ then } 1 \text{ else } 0\}$):

$$\sum_{i=1}^{N} \sum_{j \in B_i} \perp\left(\min_{\boldsymbol{u}} \|\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}|X), \boldsymbol{\xi}_i) - \boldsymbol{x}_{ij}\| \right). \quad (8)$$

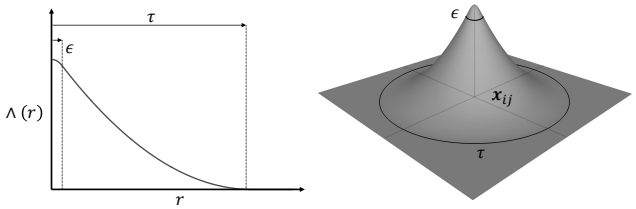


Figure 2. **Left**: Plot of the shrinking kernel $\Lambda(r)$ with respect to the reprojection error r. **Right**: 3D representation of the shrinking kernel centered at a particular background pixel $\boldsymbol{x}_{ij}$.

This transformation expresses raycasting as an optimization problem, but not an easy one: first, the L_0-like function has a zero-sized basin of convergence, and second, we cannot use the lifting trick. We rectify the first deficiency by using a more tractable proxy. A natural proxy to use for bounded terms is the L_1 proxy, but here this introduces complex bound constraints, meaning that the advantages conferred by a convex proxy are lost. We can choose as an alternative the complement of almost any flattening robust kernel, with the following desirable properties. It should be continuous and differentiable to allow the use of smooth-surface optimizers, which have been shown to provide significant improvements in convergence for the data term [18]. It also improves the efficiency of our solver (see §5) if the proxy is convex almost everywhere and flat (i.e. has a local maximum) at its peak. A suitable choice is the kernel used in the graduated non-convexity algorithm of Blake and Zisserman [2], which we name the 'shrinking kernel' (SK) to describe its effect on parts of the model that spill over the background:

$$\Lambda(r) = \begin{cases} \left(1 - \frac{\epsilon}{\tau}\right)\left(1 - \frac{r^2}{\epsilon\tau}\right) & r < \epsilon \\ \left(1 - \frac{r}{\tau}\right)^2 & \epsilon \leq r \leq \tau \\ 0 & r > \tau \end{cases} \quad (9)$$

with $\epsilon \ll \tau$ as depicted in Fig. 2. Other alternatives like a quartic polynomial could be used instead; we chose the shrinking kernel because it is the simplest one that fulfills our conditions.

We thus define our background energy by replacing $\perp$ with Λ in (8):

$$\hat{E}^b(X, \xi) = \lambda_b \sum_{i=1}^{N} \sum_{j \in B_i} \Lambda\left(\min_{\boldsymbol{u}} \|\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}|X), \boldsymbol{\xi}_i) - \boldsymbol{x}_{ij}\| \right). \quad (10)$$

We can also now easily correct the second deficiency in (8): access to lifting. By noting that Λ is monotonic, we obtain $\Lambda(\min_x f(x)) = \max_x \Lambda(f(x))$ so

$$\hat{E}^b(X, \xi) = \lambda_b \sum_{i=1}^{N} \sum_{j \in B_i} \max_{\boldsymbol{u}} \Lambda\left(\|\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}|X), \boldsymbol{\xi}_i) - \boldsymbol{x}_{ij}\| \right), \quad (11)$$

which can be subject to lifting as above (4), by defining latent variables $\mathcal{U}^b = \{\boldsymbol{u}_{ij}^b\}$ for each $i = 1 \dots N$ and $j \in B_i$:

$$E^b(X, \xi, \mathcal{U}^b) = \lambda_b \sum_{i=1}^{N} \sum_{j \in B_i} \Lambda\left(\|\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}_{ij}^b | X), \boldsymbol{\xi}_i) - \boldsymbol{x}_{ij}\|\right) \tag{12}$$

with $\hat{E}^b(X, \xi) \geq E^b(X, \xi, \mathcal{U}^b)$ for all $\mathcal{U}^b$.

4.2.1 Fixed vs adaptive τ

The value of τ in (9) significantly changes the effect of the background term (12) on the overall optimization. A high value of τ increases the number of pixels pushing the model inwards ($r \leq \tau$) and leads to a smoother energy which is easier to optimize. However, a high τ also implies that the background term competes with the data term at the object boundaries and prevents the model from fitting the data in these areas. Conversely, a low value for τ implies fewer pixels pushing inward and a sharper energy but also less competition between the background and the foreground terms. To overcome these limitations, we employ an adaptive $\tau_i(\boldsymbol{x})$ which depends on the pixel $\boldsymbol{x}$ and the image i. Thus, τ will be low for pixels close to the silhouette and will be higher otherwise. The function which measures the minimum distance from any pixel to the object silhouette is the distance transform $\mathrm{DT}_i(\boldsymbol{x}) : \mathbb{R}^2 \mapsto \mathbb{R}$. Therefore, the adaptive width of the shrinking kernel is given by $\tau_i(\boldsymbol{x}) = \min(\mathrm{DT}_i(\boldsymbol{x}), \tau_{\max})$ for a given image i. A maximum width $\tau_{\max}$ must be set because the shrinking kernel is intended to work close to the model boundaries and would be ineffective and inefficient if τ took arbitrarily high values.

This strategy takes advantage of the distance transform by using information about proximity to the silhouette to avoid pushing the model beyond it, but it flattens appropriately far from the silhouette, unlike the distance transform, whose gradients are often wrong or misleading (see §6 and Fig. 3).

5. Solver

We combine the data and background terms with regularizers $E^R(X)$ (described in Appendix A) to build the overall optimization problem:

$$\min_{X, \mathcal{U}, \xi} \left\{ E^d(X, \xi, \mathcal{U}) + \max_{\mathcal{U}^b} \left\{ E^b(X, \xi, \mathcal{U}^b) \right\} + E^R(X) \right\} \tag{13}$$

This energy is highly non-linear, non-convex and combines minimization and maximization processes that appear challenging to solve jointly. The problem contains some standard components: the combination of minimization and maximization recalls the concave-convex procedure [22] and DC programming [17]. However the natural decompositions

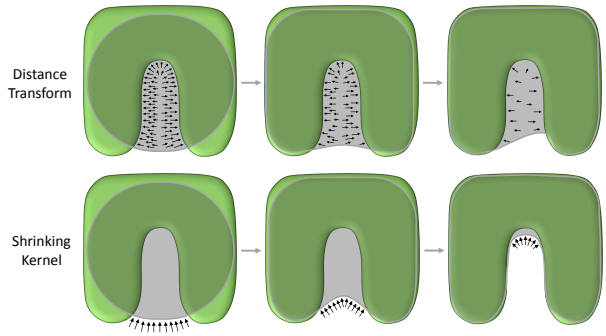


Figure 3. Evolution of the surface throughout the optimization process. The images illustrate how each background term pushes the surfaces: DT creates gradients for every sample of the surface projecting out of the silhouette. However moving surface *samples* off the background can be achieved without moving the *surface* off the background. In contrast, SK acts at every background *pixel*, but with nonzero gradient only in a thin band around the silhouette.

of our problem (i.e. the objectives of the min and max as written) are not concave so these techniques do not directly apply.

Our solver is divided into two stages: an inner maximization and an outer minimization. The inner maximization finds the background correspondences $\mathcal{U}^b$ (raycasting) that are needed for every iteration of the outer minimization problem. In turn, the outer minimization will be solved iteratively by generating updates for the control vertices X, the foreground correspondences $\mathcal{U}$ and the camera poses ξ, guaranteeing that each iteration decreases the overall energy.

5.1. Inner Raycasting Maximization

The first task is to find the correspondences for the background term, that is

$$\mathring{\mathcal{U}}^b = \arg\max_{\mathcal{U}^b} \left\{ E^b(X, \xi, \mathcal{U}^b) \right\}. \tag{14}$$

The correspondences within the vector $\mathcal{U}^b$ are independent from each other, which means that (14) can be solved with independent optimizations over Ω for each pixel. These remain nonlinear optimizations, but we can make use of Levenberg-Marquardt. Although E^b is non-quadratic, monotonicity of Λ means that $\arg\max_u \Lambda(f(u)) = \arg\min_u f(u)^2$ for a smooth function $f : \Omega \mapsto \mathbb{R}^2$. Note that this transformation applies only because the optimizations are independent per pixel: it is not the case that $\arg\max_u \Lambda(f_1(u)) + \Lambda(f_2(u)) = \arg\min_u f_1^2(u) + f_2^2(u)$ in the general case when f_1 and f_2 both depend on all of u. To improve efficiency, the background correspondences stop being updated if their projection error is higher than a given threshold $\upsilon > \tau$, thereby discarding all those pixels of the background which are too far from the model to provide any help.

In order to update these correspondences within the optimizer, correspondences may need to transition between different faces of the control mesh. As Catmull-Clark subdivision surfaces are nearly-everwhere C^2 continuous, these transitions do not harm the differentiability of our energy terms, and the correspondence updates can be handled using the strategy described in [3, 19]. Transitions of the foreground correspondences $\mathcal{U}$ in the outer minimization are handled similarly.

5.2. Outer Minimization

Now we need to solve

$$\min_{X,\mathcal{U},\xi} \{f(X,\mathcal{U},\xi) + g(X,\xi)\} \tag{15}$$

with

$$f(X,\mathcal{U},\xi) = E^d(X,\xi,\mathcal{U}) + E^R(X), \tag{16}$$

$$g(X,\xi) = E^b\left(X,\xi,\mathring{\mathcal{U}}^b(X,\xi)\right). \tag{17}$$

The Levenberg-Marquardt algorithm does not appear to be applicable here because not every term is expressed in the form of sum of squares and, moreover, the background term E^b is not convex. However, we will show that thanks to the particular choice (9) adopted to approximate the raycasting function (6), the Levenberg-Marquardt algorithm can be applied and it leads to an efficient optimization strategy. First of all, the shrinking kernel is flat at its maximum, which makes the Jacobian computation much easier, as the multiplicands of the difficult terms $\frac{\partial \mathring{\mathcal{U}}^b(X,\xi)}{\partial X}$ are zero:

$$\frac{\partial g(X,\xi)}{\partial X} = \frac{\partial E^b(X,\xi,\mathring{\mathcal{U}}^b)}{\partial X} + \cancelto{0}{\frac{\partial E^b(X,\xi,\mathring{\mathcal{U}}^b)}{\partial \mathcal{U}^b}} \cdot \frac{\partial \mathring{\mathcal{U}}^b(X,\xi)}{\partial X},$$

$$\frac{\partial g(X,\xi)}{\partial \xi} = \frac{\partial E^b(X,\xi,\mathring{\mathcal{U}}^b)}{\partial \xi} + \cancelto{0}{\frac{\partial E^b(X,\xi,\mathring{\mathcal{U}}^b)}{\partial \mathcal{U}^b}} \cdot \frac{\partial \mathring{\mathcal{U}}^b(X,\xi)}{\partial \xi}.$$

For the sake of clarity, the Jacobians have been written as scalar partial derivatives. Secondly, the shrinking kernel is defined with $\epsilon \ll \tau$ (see Fig. 2) and, hence, it is convex and can be expressed as a sum of squares almost everywhere (apart from a small area surrounding its peak). Therefore, we use all those pixels with correspondences lying in the convex area and with non-null gradient ($\epsilon \leq r \leq \tau$), and omit those which are just at the maximum or very close to it. In practice, this does not have any detrimental effect over the minimization process because this approximation discards only pixels which have a ray intersecting with the model or very close to it. Rays that intersect with the model ($r = 0$) contribute no gradient as previously shown; only a tiny fraction of the pixels discarded will have a ray which does not intersect and yet still lies in the concave area ($0 < r \leq \epsilon$).

Every iteration of the Levenberg-Marquardt algorithm involves the construction of a sparse and large linear system which is solved by applying a Cholesky LDLT decomposition. Moreover, to overcome/avoid local minima due to wrong correspondence associations, we periodically perform a global search by uniformly sampling the subdivision surface and checking for each pixel $j \in D_i$ (foreground) whether any of these samples reduces its energy E^d_{ij}. A similar search is also performed for the background correspondences.

5.3. Coarse-to-Fine

Subdivision surfaces provide a refinement relation $\mathcal{R}$ that densifies the control mesh without modifying the surface:

$$\boldsymbol{s}(\boldsymbol{u}|X) = \boldsymbol{s}(\boldsymbol{u}|\mathcal{R}X) \quad \text{for all } \boldsymbol{u} \in \Omega. \tag{18}$$

This refinement can be iterated to define a series of control meshes $X^l = \mathcal{R}^l X$, all of which represent the same limit surface. Here l denotes a given level within the coarse-to-fine scheme. This means we can optimize a coarse model for the control vertices $X = X^0$, then apply $\mathcal{R}$ to obtain a new set of model freedoms X^1 without changing E in (13). We then optimize $X = X^1$ using (13) to obtain optimal control vertices at level 1, and iterate this procedure until we find a solution at the target control mesh density. Thus, the optimizer is able to fit a detailed model with many control vertices by using the coarse model to find the energy well for a good local minimum.

6. Experiments

We conducted a series of experiments to compare our approach with the popular distance transform (DT) method for enforcing silhouette consistency [6, 21]. To perform these comparisons we implemented an alternative background term E^b_{DT} by sampling the subdivision surface uniformly at L fixed locations $\{\boldsymbol{\sigma}_l\}_{l=1}^L$ in Ω, and projecting the samples into the distance transforms of each of the N depth images:

$$E^b_{\text{DT}}(X,\xi) = \lambda_{\text{DT}} \sum_{i=1}^{N} \sum_{l=1}^{L} \text{DT}_i^2\left(\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{\sigma}_l|X), \boldsymbol{\xi}_i)\right). \tag{19}$$

We present three distinct experiments to compare our background term with the standard DT-based term (19). These experiments are intended to investigate the behaviour of our proposal in common computer vision scenarios, not to demonstrate state-of-the-art algorithms for 3D reconstruction or tracking. The first two experiments address the 3D reconstruction problem from single or multiple views respectively. In the third test we track a non-rigid object (a person) through a sequence of depth images. Moreover, we include two additional experiments in the supplementary material, together with details of the image segmentation and the computational cost of the different tested methods.

For a better visualization of the results presented here, we encourage the reader to watch the demonstration video.

6.1. Modeling an Arch

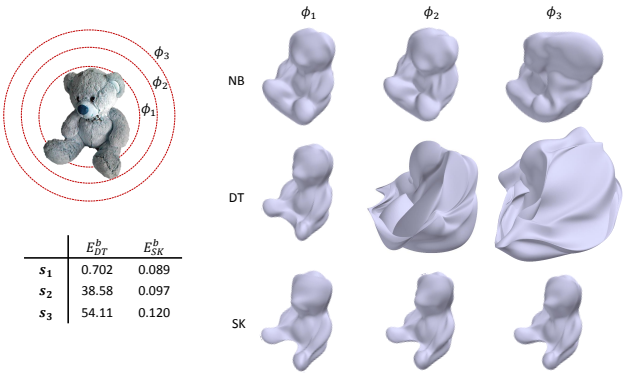
Our first experiment is a synthetic test where the data to fit consists of a single depth image generated as the front view of a smooth arch. To initialize the mesh, we compute the bounding box of the data and apply $\mathcal{R}$ twice to generate a mesh with enough degrees of freedom to deform and adapt to the data. Its corresponding initial surface is roughly an ellipsoid, which is quite far from the arch that we aim to reconstruct (see Fig. 1). In the experiment, we compare four different strategies for the background term: DT (19), SK (12) with fixed τ (SK_1), SK (12) with adaptive τ (SK_2) and without background term (NB). The weights associated to the different background terms (λ_b and λ_{DT}) are tuned so that all the background energies have the same initial value.

The final solutions are depicted in Fig. 1. We observe that the distance transform is unable to shrink the model properly while the two versions of our approach do it almost perfectly. The DT's poor behaviour has two causes. First, our formulation of E^b_{DT} implements a discrete sampling over the model instead of integrating each DT_i over all of Ω. However, the second reason for the failure is that the gradients of the DT function (shown in Fig. 3) are mostly horizontal between the two pillars of the arch. This gives no reason for the model to shrink vertically. Instead, it stretches horizontally in both directions to move the sampled model positions $\{\boldsymbol{\sigma}_l\}$ out of the penalized image region while leaving model surface stretched in-between. On the other hand, the shrinking kernel always pushes the surface in the opposite direction to the surface normals at the model silhouette. The only pixels where the shrinking kernel has non-zero gradient are those whose ray does not intersect the model but comes close to it ($\epsilon \leq d \leq \tau$). Thus, the SK background term projects the 3D model onto the image plane and pushes the surface inward on those parts of the model's silhouette that project out of the real silhouette (see Fig. 3).

The number of iterations (see demonstration video) required is significantly reduced by SK_2 over SK_1, and is similar to DT. Interestingly, SK_2 also finds a better optimum for the data term that NB is directly optimizing, by helping the model to distribute its freedoms more usefully. Although SK_1 is also able to create the gap between the columns of the arch, its energy after convergence is higher because the data and the background terms compete at the object boundaries.

6.2. Incorrect Segmentations

Another problem associated with the DT background term is the fact that an incorrect segmentation, even if there is just a single misclassified pixel, can lead to a very different distance transform which might be detrimental for the 3D reconstruction. A perfect segmentation is hard to obtain in many practical cases, so robustness to segmentation errors is important. In this section we compare basic 3D reconstructions obtained with no background term (NB), with DT and with SK (adaptive τ) when the segmentation of the object is imperfect.



	E^b_{DT}	E^b_{SK}
s_1	0.702	0.089
s_2	38.58	0.097
s_3	54.11	0.120

Figure 4. **Top left**: Illustration of the object to model together with the outline of the initial spheres used in the 3 sets of tests. **Bottom left**: Final energy terms after the optimization. **Right**: 3D reconstructions obtained without background term, with DT and with SK for the 3 different initializations ϕ_1, ϕ_2 and ϕ_3. SK is less dependent on initialization, and is better on the gap between the legs.

We test on a multi-view 3D reconstruction problem with $N = 4$ depth images of a teddy bear taken from different camera angles. We segment the depth images using background subtraction as explained in the supplementary material. Since the measurement error grows quadratically with depth, we use a comparison threshold between the background and the input images $\{Z_i\}_{i=1}^N$ that also grows quadratically with depth, in an attempt to avoid many false positive detections at distant areas. However, the resulting segmentation is still imperfect and every image contains scattered pixels or small distant regions which are mistakenly tagged as object. While it would be possible to post-process the segmentations further for better results, we leave them in this unprocessed state as our intention is to test the robustness of each method to segmentation errors.

We compare the basin of convergence for the reconstruction in each case, by starting with three different initial control meshes. These meshes are cubes placed at the centroid of the data points with edges set to 0.4, 0.5 and 0.6 meters respectively. The control meshes generate roughly spherical surfaces with diameters denoted by ϕ_1, ϕ_2 and ϕ_3 in Fig. 4. In this experiment we give high weights λ_b and λ_{DT} to the background terms to test whether the algorithm is able to shrink the model to the convex hull. The optimization is run in a coarse-to-fine scheme with 4 levels, each one running 25 iterations of LM to solve (15).

Quantitative and qualitative results are presented in Fig. 4, which shows that the data term alone is able to shrink the

model partially in the first two cases but completely fails for the largest initialization. It also fails to create the expected gap between the teddy's legs. On the other hand, DT produces a good result for s_1 but fails dramatically for s_2 and s_3 because the DT gradients far from the target object are sometimes directed towards pixels that are incorrectly segmented as data. In fact, they even force the model to protrude and deform in undesirable ways, worsening the solution compared to that without the background term. Finally, SK is able to shrink the model into the convex hull in all cases and successfully separates the teddy bear's legs. The final energies associated to the background terms are also shown in Fig. 4.

6.3. Tracking under Poor Illumination Conditions

In this last experiment we compare the two different background terms focusing on the regions $\{C_i\}$ that can be segmented as neither object nor as background. These regions are common artefacts of the capture mechanisms used by depth cameras, but the same problem could arise with RGB images if there are areas which cannot be segmented properly and remain uncertain.

The goal of the experiment is to track the body of a person who moves in front of an RGB-D camera. To better illustrate the differences between the compared methods, the sequence of images has been recorded outdoors where the depth measurements have a lower quality due to the sun's radiation. The testing sequence consists of 20 images subsampled from a longer sequence of 60 to increase the displacement between consecutive frames. Images are segmented by thresholding depth since the person is always closer to the camera than the background points. The compared background terms are configured so that they have similar weights during the optimization process. For the experiment we assume that an initial mesh (X^o in (22)) with the shape of a person is provided. Coarse-to-fine is not used here, i.e., the size and topology of this mesh does not change during the experiment.

Under the presence of invalid or uncertain measurements, we need to decide between two options to compute the distance transform. The first is to compute the distance transform for both background and invalid pixels, setting to zero only those pixels that are segmented as object ($\mathrm{DT}_i(\boldsymbol{x}_{ij}) = 0$ iff $j \in D_i$). We refer to this strategy as $\mathrm{DT}_{\mathrm{all}}$. The main disadvantage of $\mathrm{DT}_{\mathrm{all}}$ is that it shrinks the model beyond the real silhouette of the object, because the object is likely to be visible from some of those pixels tagged as invalid ($j \in C_i$). The second option, denoted here as $\mathrm{DT}_{\mathrm{safe}}$, sets to zero both the pixels observing the object and the invalid depth measurements, in an attempt to create a distance transform that only penalizes pixels that are known to observe the background ($\mathrm{DT}_i(\boldsymbol{x}_{ij}) = 0$ iff $j \notin B_i$). This alternative strategy is less restrictive and gives the model more freedom to adapt to the data properly, but it also has a drawback: the invalid area surrounding the object may have arbitrary size and contour, which can lead to gradients ∇E^b_{DT} with directions that are harmful to the model (see Fig. 5). In contrast, the shrinking kernel penalizes only background pixels (so it does not overconstrain the final solution) without being affected by the null depth measurements.

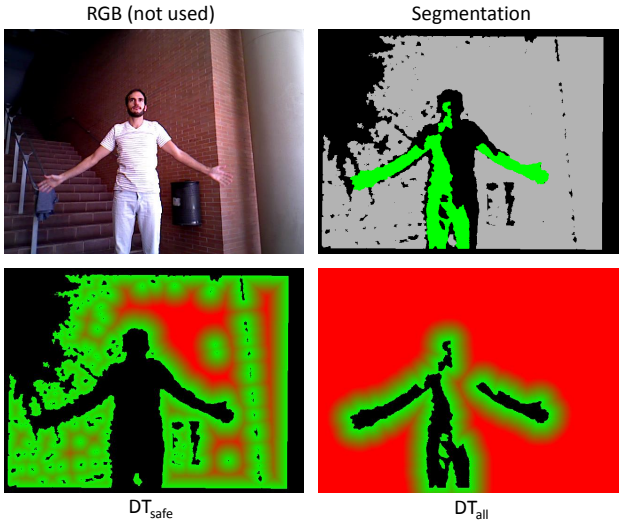


Figure 5. Differences between $\mathrm{DT}_{\mathrm{all}}$ and $\mathrm{DT}_{\mathrm{safe}}$ for one of the frames used for tracking. **Top left**: RGB image included for clarity but not used in the algorithm. **Top right**: Segmented image. The foreground pixels are shown in green, background in gray and null depth in black. **Bottom**: $\mathrm{DT}_{\mathrm{all}}$ and $\mathrm{DT}_{\mathrm{safe}}$ starting from zero (green) and truncated at 50 (red) for a better visualization.

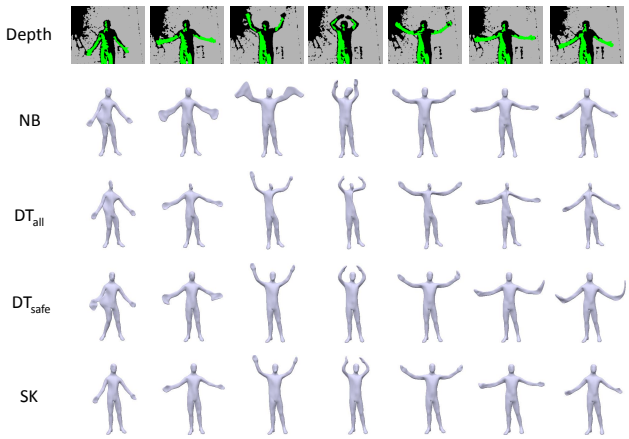


Figure 6. **Top row**: Some of the depth images used for tracking. Black represents null depth, pixels in gray observe the background and those in purple the object. **Rows 2-5**: Results after convergence for the selected images and the different tested approaches.

Qualitative results are shown in Fig. 6. It can be observed that, in the absence of a background term, the model fits the data but sometimes protrudes out of the silhouette. Moreover, it allows for wrong correspondences between the model

and the data, as occurs sometimes for the head (middle column) which tries to fit some of the arm points. DT_{all} provides the worst results because it tries to push the model out of the areas with null depth, and almost half of the pixels observing the person have null depth. DT_{safe} performs better but still leads to some artefacts as the gradients of the distance transform are not always directed towards the target. Finally, SK achieves the best results, keeping the surface within the silhouette during the whole sequence without leading to artefacts or protusion of the surface.

The same experiment has been carried out indoors under good illumination conditions. In that case, the number of pixels with null depth is much lower and both DT and SK provide equally good results.

7. Conclusions

This paper describes a novel background term that forces a 3D model to shrink within the visual hull of an object observed from one or multiple views. To demonstrate its superior performance over the popular distance transform-based formulation, we introduced a unified framework to address the problems of 3D reconstruction or non-rigid tracking with smooth surface models. Results demonstrate that our proposal enforces silhouette consistency more effectively than the distance transform. Specifically, it works better with real data that often include noise and uncertainty and which cannot always be segmented perfectly. This proposal could therefore be extended to RGB-based reconstruction and tracking systems.

Future work includes finding a better solver for this concave-convex optimization problem, which would optimize all variables jointly, and adapting the topology of the mesh during optimization.

A. Regularization

The data and background terms (4) and (12) guarantee that the model fits the data and keeps within the convex hull of the object. However, the solution found using these terms alone can include creases and sharp edges that make the 3D model unappealing. The mesh can also degenerate throughout the optimization process, leading to ill-posed configurations that eventually cause the 3D reconstruction or the tracking system to fail. For these reasons, we introduce two regularization terms that encode a smoothness prior on the object we are trying to reconstruct. Moreover, for the tracking problem, we include other two extra terms to keep the subdivision surface as rigid as possible while tracking the target object. The sum of these terms forms $E^R(X)$.

To keep the surface smooth we penalize the **gradient of the surface normals**, a proxy for surface curvature. We approximate this using a discrete sum by homogeneously sampling the subdivision surface over its parametric domain $\square \times \{1..F\}$ to obtain F sets of K samples per face, denoted $\boldsymbol{s}_{fk}$. Then the surface smoothness regularizer is

$$E^s(X) = \lambda_s \sum_{f=1}^{F} \sum_{k=1}^{K} \frac{\left\|\nabla_{u_x} \boldsymbol{s}^{\perp}(\boldsymbol{s}_{fk}|X)\right\|^2}{\left\|\nabla_{u_x} \boldsymbol{s}(\boldsymbol{s}_{fk}|X)\right\|^2} + \frac{\left\|\nabla_{u_y} \boldsymbol{s}^{\perp}(\boldsymbol{s}_{fk}|X)\right\|^2}{\left\|\nabla_{u_y} \boldsymbol{s}(\boldsymbol{s}_{fk}|X)\right\|^2}, \tag{20}$$

where we use forward finite differences to approximate the gradients $\nabla \boldsymbol{s}^{\perp}(u) \in \mathbb{R}^3$ and $\nabla \boldsymbol{s}(u) \in \mathbb{R}^3$.

In addition, we want control vertices to be spread as evenly as possible over the model, so we must avoid the control mesh from stretching and distorting arbitrarily. We enforce this by adding a simplified and discrete version of the **membrane energy**. If $\boldsymbol{e}_{f,k}$ denotes the kth edge of the quadrilateral face f, this second regularization term is defined as

$$E^h(X) = \lambda_r \sum_{f=1}^{F} \sum_{k=1}^{4} \left\|\boldsymbol{e}_{f,k}(X)\right\|^2. \tag{21}$$

E^h penalizes long edges and indirectly favours isometry.

When the goal is to track a non-rigid object, we assume that a mesh with the shape of the object is given. This mesh moves and deforms over time to fit the new incoming data but, at the same, it must keep its original proportions. To enforce such behaviour, we include the **as-rigid-as-possible** regularizer (ARAP) [16]:

$$E^a(X, \zeta) = \lambda_a \sum_{v=1}^{P} \sum_{k \in \mathcal{N}_v} \left\| R(\boldsymbol{\zeta}_v)(\boldsymbol{X}_v^o - \boldsymbol{X}_k^o) - (\boldsymbol{X}_v - \boldsymbol{X}_k) \right\|^2 \tag{22}$$

where X^o gives the initial locations of the control vertices of the mesh employed for tracking, and $\mathcal{N}_v$ contains the neighbours for vertex v. For each control vertex, the relative rotation of the mesh with respect to its original configuration is represented by the matrix $R(\boldsymbol{\zeta}_v) \in \text{SO(3)}$. These rotations are minimized, and are regularized so that the deformation of the mesh is locally smooth:

$$E^r(\zeta) = \lambda_r \sum_{v=1}^{P} \sum_{k \in \mathcal{N}_v} \left\|\boldsymbol{\zeta}_v - \boldsymbol{\zeta}_k\right\|^2 \tag{23}$$

The minimization is written $E^R(X) = \min_{\zeta} E^R(X, \zeta)$, and the rotations ζ are lifted into the overall problem.

References

[1] P. Besl and N. Mckay. A method for registration of 3-D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 1

[2] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, USA, Aug. 1987. 3

[3] T. Cashman and A. Fitzgibbon. What shape are dolphins? Building 3D morphable models from 2D images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(1):232–244, 2013. 1, 2, 3, 5

[4] K.-S. Cheng, W. Wang, H. Qin, K.-Y. Wong, H. Yang, and Y. Liu. Fitting subdivision surfaces to unorganized point data using SDM. In *12th Pacific Conference on Computer Graphics and Applications*, pages 16–24, 2004. 2

[5] D. Cremers and K. Kolev. Multiview stereo and silhouette consistency via convex functionals over convex domains. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(6):1161–1174, 2011. 2

[6] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real-time human pose tracking from range data. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 738–751, 2012. 1, 2, 5

[7] W. Gander, G. H. Golub, and R. Strebel. Least-squares fitting of circles and ellipses. *BIT Numerical Mathematics*, 34(4):558–578, 1994. 1, 3

[8] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proc. SIGGRAPH*, pages 295–302, 1994. 2

[9] S. Ilic. Using subdivision surfaces for 3-D reconstruction from noisy data. In *Workshop on Image Registration in Deformable Environments (DEFORM)*, pages 1–10, 2006. 2

[10] S. Ivekovic and E. Trucco. Fitting subdivision surface models to noisy and incomplete 3-D data. In *Proc. Computer Vision/Computer Graphics Collaboration Techniques: MIRAGE*, pages 542–554, 2007. 2

[11] N. Litke, A. Levin, and P. Schröder. Fitting subdivision surfaces. In *Proc. Conf. on Visualization*, pages 319–324, 2001. 2

[12] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Int. Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011. 2

[13] M. R. Oswald, E. Töppe, and D. Cremers. Fast and globally optimal single view reconstruction of curved objects. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 534–541, 2012. 1, 2

[14] M. Prasad, A. Zisserman, and A. W. Fitzgibbon. Single view reconstruction of curved surfaces. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1345–1354, 2006. 2

[15] N. Savinov, L. Ladicky, C. Hane, and M. Pollefeys. Discrete optimization of ray potentials for semantic 3D reconstruction. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5511–5518, 2015. 2

[16] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, 2007. 8

[17] P. D. Tao and L. T. H. An. Convex analysis approach to DC programming: Theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1):289–355, 1997. 4

[18] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, et al. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics (TOG)*, 35(4):143, 2016. 2, 3

[19] J. Taylor, R. Stebbing, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Hertzmann, and A. Fitzgibbon. User-specific hand modeling from monocular depth sequences. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 644–651, 2014. 2, 3, 5

[20] E. Töppe, M. R. Oswald, D. Cremers, and C. Rother. Image-based 3D modeling via Cheeger sets. In *Proc. Asian Conf. on Computer Vision (ACCV)*, pages 53–64. 2010. 2

[21] S. Vicente and L. Agapito. Balloon shapes: Reconstructing and deforming objects with volume from images. In *Proc. Int. Conf. on 3D Vision (3DV)*, pages 223–230, 2013. 2, 5

[22] A. L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). In *Advances in neural information processing systems (NIPS)*, volume 2, pages 1033–1040, 2002. 4

Supplementary Material: An Efficient Background Term for 3D Reconstruction and Tracking with Smooth Surface Models

Mariano Jaimez[1,2] Thomas J. Cashman[3] Andrew Fitzgibbon[3]
Javier Gonzalez-Jimenez[1] Daniel Cremers[2]

[1]University of Malaga [2]Technical University of Munich [3]Microsoft

This document details and extends the experimental section presented in the main manuscript. We describe how the depth images are segmented into foreground-background regions and include additional experiments to compare our new background term (SK) with the DT-based formulation.

Two new experiments are presented. The first one analyses the behaviour of the different tested methods when the data to fit gets significantly far from the initial mesh used for tracking. The second one addresses the problem of 3D reconstruction from depth images with wrong/perturbed initial camera poses. Last, the temporal performance of the different methods is also evaluated.

1. Experiment Initialization and Segmentation

For each experiment, each of the N images captured by the depth camera already provides the set of invalid depth pixels C_i. We have implemented three different approaches to segment the remaining pixels into the regions corresponding to object (D_i) and background (B_i):

- Segmentation by plane removal. The object is placed on a flat surface and the camera must mostly observe the object and the plane it is lying on. B_i is defined by proximity to a plane fitted to the flat surface and D_i by the remainder.
- Segmentation by background subtraction. The camera pose is fixed and several images of the object are taken. By capturing a single image of the background without the target object present, we can define D_i using those pixels that change between the former images and the background image.
- Segmentation by depth thresholding. D_i is simply defined to be those pixels within a depth range previously set, and B_i to be the remainder.

Moreover, our algorithm requires an initial guess for the camera poses, which does not need to be very accurate and we assume is provided by the user.

2. Tracking distant data

In this experiment we evaluate the performance of the compared methods when the distance between the data to fit and the mesh increases. To that end we have recorded an RGB-D sequence of a person moving his arms, in this case indoor and with good visibility conditions. Instead of tracking consecutive images as in §6.3 of the main manuscript, the initial mesh is aligned with the data of each image independently to evaluate how the different background terms help the model to converge to the right pose. Apart from that aspect, the procedure is similar to the one described in §6.3 of the main manuscript: an initial mesh is provided and the optimization problem is solved directly for this mesh without resorting to coarse-to-fine.

Six different images with a resolution of 120×160 are considered for the experiment. Results are shown in Fig. 1. It can be noticed that NB works well for the first three images where data is closer to the initial mesh, but fails to push the arms forward for the last images and wrongly deforms the main body to fit points which actually correspond to the left arm. $\mathrm{DT}_{\mathrm{all}}$ provides the best results since in this case the segmentations are almost perfect and the number of pixels with null depth around the person is quite low. Hence, the DT gradients help the model to converge to the right solution very quickly. On the other hand, $\mathrm{DT}_{\mathrm{safe}}$ provides results which are even worse than those of NB because some image borders have null depth and therefore attract the arms towards them. In this particular example the testing images could be processed to correct this deficiency but this is not possible in general (see the sequence in §6.3 of the main manuscript) and, therefore, we keep the original images to illustrate the drawbacks of this approach. Last, SK provides accurate results but for the fifth image. This is a very challenging experiment for SK because it pushes the model inwards along its silhouette which, in this case, mostly leads to compressing the person's arms. Only the fingertips provide helpful gradients to bring the arms towards the main body. Despite this fact, SK outperforms NB and only fails for the fifth image where a small number of points

Figure 1. Results after running the optimization problem to align a given initial mesh of a person with geometric data associated to different postures. The first tested images (left columns) observe a person whose posture is close to the initial mesh while the person's posture in the the last ones (right columns) is considerably different from that of the initial mesh.

of the almost-hidden arm causes the model to deform in an undesirable way (and the optimization to fall into a bad local minimum).

3. Robustness to Wrong Initial Camera Poses

For this experiment we have recorded a continuous RGB-D sequence by moving a handheld camera around a teddy bear. In order to get accurate camera poses we have run a voxel-based SLAM method which combines [1] and [2]. The purpose of this experiment is to test the basin of convergence of the different approaches (here SK, $\mathrm{DT}_{\mathrm{all}}$ and $\mathrm{DT}_{\mathrm{safe}}$) when the camera poses are not initialized correctly. To that end, we will consider the pose estimates provided by [1] as ground truth and will generate perturbed initial camera poses by adding Gaussian noise to them (the ground truth poses are actually not error-free but they are precise enough for the evaluation). We decimate the sequence and retain only four depth images to address the 3D reconstruction problem. To be able to measure the deviations with respect to the original camera poses, we fix the first camera and only perturb and optimize for the other three.

The image resolution employed is 60×80, and the optimization runs until convergence within a coarse-to-fine scheme. The initial control mesh is obtained as the bounding box of the data, and is refined once (applying $\mathcal{R}$) before starting the optimization process. For this experiment we employ three coarse-to-fine levels although we run the optimization for the first level twice: first with strong regularization to avoid the mesh to deform too much while the camera poses are far from their true positions, and second with a normal weighting to allow the surface to fit data. Since the teddy bear was lying on a table when the sequence was recorded, the segmentations can be obtained by plane removal.

We run a total of 50 tests for each method; results are shown in Fig. 2. SK provides the lowest error on average since it is able to recover the camera poses and shrink the model without penalizing the data term. On the other hand, $\mathrm{DT}_{\mathrm{all}}$ is always able to bring the cameras to their right configuration but it goes too far by pushing the model inwards when it projects on C_i (null depth), leading to higher average pose errors. Last, $\mathrm{DT}_{\mathrm{safe}}$ shows an erratic behaviour, being sometimes able to bring the cameras to their right poses and sometimes failing dramatically due to the wrong DT gradients.

4. Computational Performance

All the experiments have been run on a single core of an Intel Xeon E5630 CPU at 2.53 GHz (compiled for 32 bits and running on Windows 10). We have chosen to analyse the runtimes of the experiment described in §6.3 of the main manuscript where coarse-to-fine is not used. In this case a mesh with 1128 vertices must fit the data contained in consecutive depth images with QQVGA resolution (120×160). The time taken by a complete iteration of the Levenberg-Marquardt algorithm is measured for the three cases considered: fitting without background term (NB),

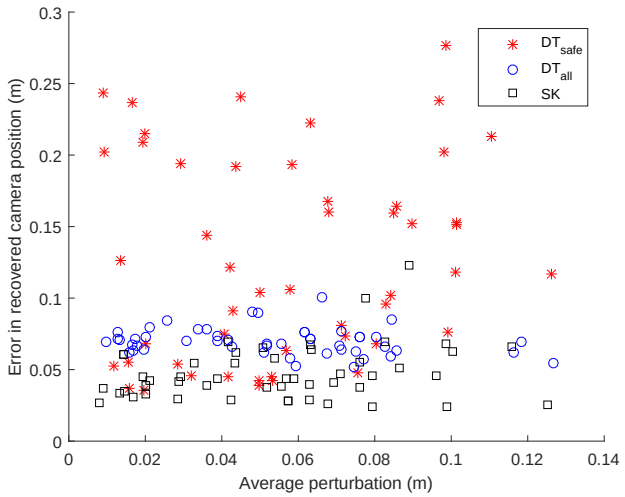


Figure 2. Error in the estimation of the camera positions as a function of the initial average perturbations. For simplicity, only the translational component of $\boldsymbol{\xi}_i$ is considered for both the perturbations and the measured errors.

with the DT-based background term and with our approach (SK). We do not include the time associated to the complete search because this step is only executed occasionally and is not part of the LM solver.

As expected, the fastest method is NB whose iterations take on average 1.46 seconds. When the background terms are included, the average runtime of the LM iterations increases up to 1.62 seconds (DT) and 2.91 seconds (SK). As expected, our proposal is the computationally heaviest alternative because it includes an inner maximization process (raycasting) within the overall optimization.

References

[1] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3D reconstruction using signed distance functions. In *Robotics: Science and Systems*, 2013. 2

[2] R. Maier, J. Stückler, and D. Cremers. Super-resolution keyframe fusion for 3D modeling with high-quality textures. In *International Conference on 3D Vision (3DV)*, pages 536–544, 2015. 2