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),

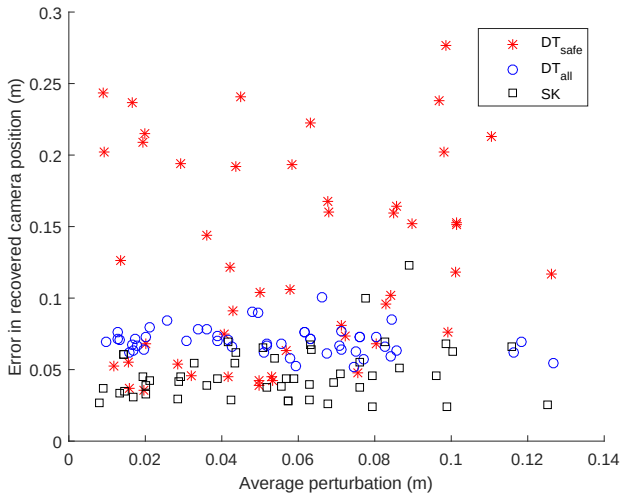

Figure 2. Error in the estimation of the camera positions as a function of the initial average perturbations. For simplicity, only the translational component of $\boldsymbol{\xi}_i$ is considered for both the perturbations and the measured errors.

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