In this paper, we present a method that allows simultaneous surface reconstruction and camera localization from monocular images in static scenes. The novel aspect of the method is its independence from any explicit feature detection schemes. Instead, it uses method similar to intensity-based bundle adjustment. Thus, it is better suited for 3D reconstruction of weakly textured surfaces.

A number of methods with similar functionality have already been described [4, 6]. All of these methods, however, rely on some kind of feature detection schemes, such as such as SIFT [2] features, FAST corner detection [8], and so on.

The basic concept of our algorithm can be summarized as follows: In traditional bundle adjustment, coordinates of 3D points that are associated with feature points are recovered from a set of 2D feature position measurements. This approach will obviously work only if a feature detecting scheme can be used at all. In our case, we do not assume that robust feature extraction is possible, and thus we do not work with 2D positions, but with image intensities.

Originally, our method was inspired by a stereo disparity tracking method developed by Ramey [7]. The generalization that we are suggesting leads to an optimization problem that corresponds to intensity-based bundle-adjustment that is restricted to two frames. Thus, our solution shares some characteristics with typical bundle-adjustment algorithms [1, 9].

Our method establishes a depth map of the region of interest within a template image that has been chosen by the user. That depth map is then a function $S_d(u, v)$ mapping a $k$-dimensional parameter vector $d$ together with image coordinates $(u, v) \in \mathbb{R}^2$ to a depth value $\lambda \in \mathbb{R}$ at the specified coordinate. Given intrinsic camera parameters, this depth map can actually be interpreted as a 3D surface. Let

- $d_n$ denote the $k$-dimensional vector of parameters of the model describing the depth map.
- $S_d(u, v)$ denote a function of type $\mathbb{R}^k \times \mathbb{R}^2 \rightarrow \mathbb{R}$ that maps model parameters together with image pixel coordinates to 1D pixel depth values.
- $p_n = (t_n, q_n)$ denote the extrinsic camera parameters corresponding to image $n$, consisting of translation vector $t_n \in \mathbb{R}^3$ and rotation quaternion $q_n \in \mathbb{R}^4$.
- $T(t, q, p) : \mathbb{R}^3 \times \mathbb{R}^4 \times \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is a transformation mapping 3D spatial coordinates $p$ to 3D coordinates in the camera frame described by a translation vector $t$ and a rotation quaternion $q$.
- $\pi(p)$ be the projection of a 3D point $p$ to 2D image coordinates, according to the internal camera calibration parameters of the camera used.
- $I_n(u, v)$ be the image function of image $n$, containing all pixel values. $I_0$ is hence the reference image function.
- $(u_1, v_1), \ldots, (u_m, v_m)$ denote the pixel coordinates of the $m$ reference pixels, chosen from the ROI in the reference image.

The problem of determining surface parameters and camera position can then be stated as minimization problem for the following objective function $o(d, p_n)$:

$$o(d, p_n) = \sum_{i=1}^{m} (c(I_0(\pi(T(p_n, r_{ui, vi}(S_d(u_i, v_i)))) - I_0(u_i, v_i)))^2 \tag{1}$$

In other words, we are seeking a function that warps image coordinates from the reference image $I_0$ to the current image $I_n$ such that intensity differences are minimized. The function $c$ is a cost function that should be chosen to be robust against outliers, such as the Pseudo-Huber [1, p. 619] cost function.

It is clear that, to actually recover the model parameters from the scene, we need some method to minimize the cost function described above. Since we are dealing with a constrained problem, an adequate method for optimization is Sequential Quadratic Programming (SQP). For a more detailed description of the method, the reader is referred to [5].

The approach as described so far worked well in situations where camera movement is sufficiently smooth and no large pixel displacements occur between subsequent frames. However, problems occurred when that was not the case. This was to be expected, since the algorithm operates on intensity values and will have trouble aligning with the correct values again if they are too far away.

We addressed this problem using Lucas-Kanade optical flow [3] to determine the camera position in a separate optimization step before the intensity-based optimization is taking place:

$$o'(p_n) = \sum_{i=1}^{m} c'(I_0(\pi(T(p_n, r_{ui, vi}(S_d(u_i, v_i)))) - I_0(u_i, v_i))) \tag{2}$$

Thus, we can summarize our strategy as using optic flow for bridging large gaps in camera movement, while intensity-based optimization refines the model and camera parameters and essentially prevents drifting away from the original point intensity values. This could easily occur if only optical flow based optimization would be performed.


