Instant Outdoor Localization and SLAM Initialization from 2.5D Maps

Clemens Arth*, *Member, IEEE*, Christian Pirchheim*, *Student Member, IEEE*, Jonathan Ventura, *Member, IEEE*, Dieter Schmalstieg, *Senior Member, IEEE*, and Vincent Lepetit

(a) (b) (c) (d)

Fig. 1: Outdoor urban usage of real-time SLAM with our novel localization technique. **(a)** User with a mobile device running our software. **(b)** Map view with sensor pose (**blue**) and the pose estimate from our method (**green**). **(c)** Reprojection of a globally aligned building model into the image using the sensor pose (**blue**) and the same reprojection after correction with our method (**green**). **(d)** Live camera view on the mobile device with globally aligned augmentations. We use the corrected pose to synthetically render a depth map from the building model and initialize a SLAM system, which starts tracking the camera motion. We can then instantly augment the scene with virtual elements. In contrast to previous systems, we do not need large translations for initializing the SLAM system, and we can handle both purely rotational and general 3D motions.

Abstract—
We present a method for large-scale geo-localization and global tracking of mobile devices in urban outdoor environments. In contrast to existing methods, we instantaneously initialize and globally register a SLAM map by localizing the first keyframe with respect to widely available untextured 2.5D maps. Given a single image frame and a coarse sensor pose prior, our localization method estimates the absolute camera orientation from straight line segments and the translation by aligning the city map model with a semantic segmentation of the image. We use the resulting 6DOF pose, together with information inferred from the city map model, to reliably initialize and extend a 3D SLAM map in a global coordinate system, applying a model-supported SLAM mapping approach. We show the robustness and accuracy of our localization approach on a challenging dataset, and demonstrate unconstrained global SLAM mapping and tracking of arbitrary camera motion on several sequences.

Index Terms—SLAM, geo-localization, image registration, 2D map, outdoor augmented reality

1 Introduction

Simultaneous Localization and Mapping (SLAM) algorithms have now been developed to an extent that allows bringing them from small indoor workspaces to challenging outdoor environments. However, SLAM only provides relative poses in an arbitrary coordinate system with unknown scale. This is clearly not enough for many AR applications such as navigation [17] or labeling of local touristic landmarks [41]. Consequently, methods have been introduced that align the local coordinate system of a SLAM map with the global coordinate system of a 3D map with metric scale [38, 23]. However, this requires the user to wait until the SLAM system has acquired a sufficient number of images to initialize its 3D map, which is not acceptable for an interactive application. Moreover, SLAM systems require specific motions for initialization, which are not easily performed by a non-expert.

- *Clemens Arth, Christian Pirchheim, Dieter Schmalstieg and Vincent Lepetit are with the Graz University of Technology, Austria. E-mail: {arth,pirchheim,schmalstieg,lepetit}@icg.tugraz.at.*
- *Jonathan Ventura is with the University of Colorado Colorado Springs. E-mail: jventura@uccs.edu*
- **The first and the second author contributed equally to this work.*

Manuscript received 18 Sept. 2014; accepted 10 Jan. 2015. Date of Publication 20 Jan. 2015; date of current version 23 Mar. 2015. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.

In parallel to the developments in SLAM, tremendous progress has been made in geo-localization of mobile devices from single frames over the last years. Since device sensors are typically not accurate enough, computer vision techniques are useful to improve their estimates. However, current methods still do not fulfill the requirements of outdoor[1] Augmented Reality (AR), namely, to provide large-scale, low-latency, robust and accurate camera registration in a global coordinate system. Some methods compute the global camera pose by matching one or more input images with a large database of pre-registered images [32, 31]. Such databases are tedious to create and maintain, and often only available for popular places. More recently, image registration methods have been presented which leverage widely available untextured 2D and 3D models of outdoor environments. These methods match input images with globally aligned 2D cadastral maps [7, 5, 8], digital elevation models [2, 4], or 2.5D models [25, 36], that can for example be obtained for free under an open license from OpenStreetMap [2].

In this paper, we introduce a novel method for instant geo-localization of the video stream captured by a mobile device (Fig. 1). The first stage of our method registers the image to an untextured 2.5D map (2D building footprints + approximate building height), providing an accurately and globally aligned pose. This is done by first estimating the absolute camera orientation from straight-line segments,

[1] When we refer to *outdoor environments*, we more precisely mean *urban outdoor environments*.

[2] OpenStreetMap: `http://www.openstreetmap.org`

then estimating the camera translation by segmenting the façades in the input image and matching them with those of the map. The resulting pose is suitable to initialize a SLAM system. The SLAM map is initialized by back-projecting the feature points onto synthetic depth images rendered from the augmented 2.5D map.

Our system has several major advantages over the state-of-the-art: First, the global localization component requires only OpenStreetMap-style data, which is widely available. Second, the global localization does not require searching through a large database and is therefore suitable for mobile devices with limited memory and computational capacity[3]. Third, the initialization of the SLAM system from the first frame avoids the need to tediously cover a sufficient outdoor baseline for stereo triangulation. Fourth, there is no restriction on the camera motion: Tracking and mapping are possible even in the case of purely rotational motion. This combination of features brings outdoor urban tracking a significant step closer to use in practice.

2 Related work

The literature on outdoor camera registration is large, and an exhaustive discussion is out of scope of the paper. In this section, we aim at providing a comprehensive overview of approaches competing with ours in delivering global outdoor localization with 6DOF.

2.1 Image-based localization

For more than a decade, image-based localization [28] has been a topic of intensive research. Given one or more input images and optionally a sensor prior (location from GPS, orientation from compass and magnetometer), these systems retrieve similar pre-registered images from a database to compute the 3D place or full 6D pose of the input image. For example, Schindler *et al*. [32] demonstrated image-based localization using databases that contain 20 km of urban street-side imagery, organized in a vocabulary tree to handle the massive amounts of data. Later works such as Sattler *et al*. [31] improved upon both accuracy and performance.

Zamir *et al*. [42] and later Vaca-Castano *et al*. [37] showed it was possible to use existing image collections, such as Google StreetView, to get an estimate of the camera pose. However, image databases such as Google StreetView or Microsoft StreetSide are not universally suited for localization, since they are only sparsely sampled, and not available at all in certain regions and countries.

The major disadvantages of image-based localization approaches remain that they do not scale well. Many images need to be captured for each new location, and, even with sufficiently dense sampling, it is still very challenging to match images under changing conditions due to illumination, season, construction activity and many other sources of change.

2.2 Localization using untextured models

Another class of localization approaches avoids image databases altogether by relying on *untextured* models such as widely available 2D cadastral maps annotated with per-building height information or digital elevation models (often from LIDAR). These methods, which seek to register 2.5D maps to single or multi-view images, are most closely related to the work presented in this paper. However, global registration from sparse 2.5D information is difficult, and we will explain the restrictions made by previous work in this area.

LIDAR images provide a relatively dense sampling of the environment compared to cadastral maps. This allows Mastin *et al*. [20] to use a mutual information approach for registering aerial images to a 2.5D model. This kind of approach is not feasible for our use case of registering street-level images, which show only a few façades, with sparse 2.5D cadastral maps.

Meierhold *et al*. [22] establish line correspondences between image and 2.5D model. They state that "the accuracy of 3D lines obtained with the presented algorithm is insufficient for the purpose of image orientation" and, therefore, suggest to resolve this problem with user interaction. Similarly, Matei *et al*. [21] manually annotate the building outlines in the input image and then match them with the model. These approaches are not applicable to AR. In contrast, our algorithm runs fully automatic without any user intervention.

Baatz *et al*. [2] use contour matching and refinement of sky silhouettes between a digital elevation model and mountain images. Similarly, Bansal *et al*. [4] verify pose hypotheses by matching the image skyline with the model. In AR, users should not have to point the camera at the skyline. In dense urban areas, the skyline may not even be easily visible. Therefore, we do not rely on a visible skyline in our input images.

Another work from Baatz *et al*. [3] registers semantically labeled images with respect to labeled 3D digital terrain models. The idea of semantic segmentation is related to our method, but Baatz *et al*. process landscape images, whereas we process urban images. They state that "it is still extremely challenging to accurately identify the semantics" and restrict themselves to four of the easier classes (sky, water, settlements, other). This approach allows Baatz *et al*. to estimate only the orientation of the image with respect to the model. In contrast, our method computes both absolute orientation and 3D location.

Several methods [8, 5, 7] use 2D cadastral maps. David *et al*. [8] register panoramic images with 2D maps using a building façade orientation descriptor. As shown by Arth *et al*. [1], the higher amount information contained in wide field-of-view images significantly increases the success rate of localization. Mobile devices have a narrow field of view, and a descriptor such as the one used by David *et al*. is not discriminant enough in such situations. Cham *et al*. [5] also consider panoramic images and aim to detect vertical building outlines and façade normals resulting in 2D fragments which are then matched with a 2D map. Chu *et al*. [7] report to outperform Cham *et al*. [5]. They compute a descriptor from vertical building outlines in perspective input images, which is then matched with a 2D map. Chu *et al*. published their code and dataset online. Upon careful analysis, the dataset consist of only 11 scenes that mostly show free-standing buildings with rectangular footprint. Matching requires the exact detection of the left, middle and right building outline in the input image. To facilitate the detection of vertical edge and vanishing points, they partially used manual annotation of the input images. In contrast, we aim at a fully automatic method.

We share the assumption of horizontal and vertical image lines with Chu *et al*. [7]. In contrast, however, we use a robust algorithm for orientation estimation, and consider a large number of potential vertical building outlines. The resulting pose hypotheses are verified based on a semantic segmentation of the image, which adds another layer of information to the pose estimation process. This allows our method to be applied to much more complex images compared to [7].

Similar to our approach, 2.5D models are employed by [25, 36]. Ramalingam *et al*. [25] establish the registration between an image and a 2.5D model by computing 3D-2D line and point correspondences. However, an already registered second image is required to establish the 3D-2D correspondences by first matching 2D-2D SIFT features between the two images. Consequently, the first image of a sequence needs to be manually annotated. In contrast, our methods are fully automatic and only require a single input image.

We share the idea of using semantic image segmentation as input for the registration with a 2.5D map with Taneja *et al*. [36]. However, Taneja *et al*. optimize the pose over a continuous 6D cost space, while we are verifying pose hypotheses at discrete positions within a 2D cost space, which makes our method arguably much faster. Taneja *et al*. use comparably detailed 2.5D models and Google StreetView images for their queries, which are high-resolution panoramas with a rather accurate initial geo-location. The quality of this data likely allows them to optimize all 6DOF simultaneously without getting stuck in local minima. With narrow field-of-view images from a mobile device annotated with noisy sensor priors, often showing only a single façade, even our 2D cost space has many local minima, making an optimization along the lines of Taneja *et al*. problematic.

[3] Note that our current unoptimized, single-threaded Matlab implementation of the global localization component is one order of magnitude away from real-time operation. However, the SLAM component runs at 30Hz, and we discuss later how the localization can be accelerated to meet real-time requirements.

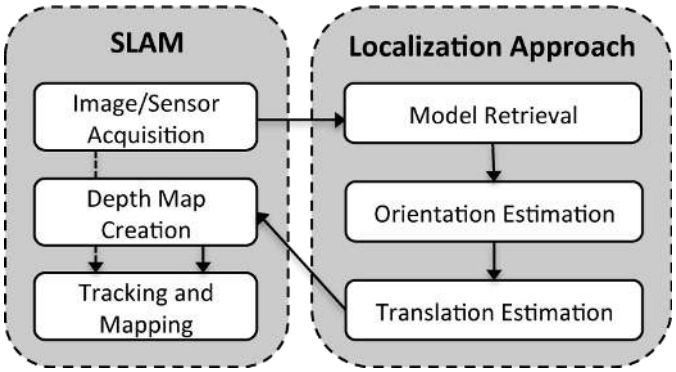

Fig. 2: An overview of our method. The SLAM component provides image and sensor data to the localization module, returning an accurate pose estimate for depth map creation and SLAM initialization.

2.3 Outdoor augmented reality tracking

Early outdoor Augmented Reality systems only used GPS and compass sensors for localization and tracking [11]. Since these approaches were not accurate enough for precise 3D augmentations, later systems fused image and sensor information (including inertial sensors) to perform tracking from photo-textured models in urban environments [15, 26]. However, both systems needed to be manually initialized, *e.g.*, by starting off from a known location in the environment. Later on, Reitmayr and Drummond [27] extended their system with a dedicated initialization procedure based on GPS sensor priors.

Most recently, SLAM systems have been used in outdoor augmented reality. SLAM does not require a model, but is capable of mapping and tracking arbitrary scenes. However, the unknown scale of local SLAM maps only allows for tracking "relative" poses. Ventura *et al*. [38] presented a localization method to align a local SLAM 3D map with a globally registered 3D map over time, and showed real-time SLAM on a mobile devices with globally registered 6DOF tracking in urban outdoor environments. A similar system was concurrently developed by Middelberg *et al*. [23]. However, before global localization can take place, a local 3D SLAM map needs to be initialized from a stereo image pair, which requires a translational camera motion. In practice, this requires the user to walk several meters to span the required baseline. In contrast, we show how to use the first keyframe acquired by the SLAM system to perform localization.

In contrast to pure model-based tracking and regular SLAM mapping, we propose to leverage untextured 2.5D models for model-assisted SLAM mapping such that the SLAM system can track absolute 6DOF poses with respect to a globally aligned SLAM 3D map.

3 Method Overview

As depicted in Fig. 2, our method first obtains a single image (camera) and a pose estimate from mobile sensors (GPS, compass, accelerometer), *i.e.*, the first keyframe acquired with a SLAM system running on a mobile device, plus a record of the built-in sensor values. From the sensor data, a first 6DOF pose estimate is compiled, using the fused compass and accelerometer input to provide a full 3×3 rotation matrix w.r.t. north/east and the earth center, and augmenting it with the WGS84 GPS information in metric UTM[4] coordinates to create a 3×4 pose matrix. This estimate is used to retrieve a 2.5D map containing the surrounding buildings, *i.e.*, using 2D building and height data from OpenStreetMap. Note that, however, that this pose can be off by 30° and 15 *m*, which is not accurate enough for SLAM initialization.

For our localization method, we assume that most image line segments extracted from the visible building façades are either horizontal or vertical. This is a common assumption used in vanishing point and relative orientation estimation, valid for many urban environments. Based on this assumption, we essentially solve a 2D-3D line correspondence problem. Already, three correct image-model correspondences allow for computing the 6DOF pose. However, our correspondence problem is profoundly non-trivial, since we find very little matching information in our input data. Additionally, the pose prior provided by the mobile sensors can be very noisy.

For the estimation of the global camera orientation (Section 4), we require a single correspondence between a horizontal image line and a model façade plane. This problem is robustly solved using minimal solvers in RANSAC frameworks.

For the subsequent estimation of the global 3D camera location (Section 5), we require two correspondences between vertical image lines and model façade outlines. This problem is tackled by first extracting potential vertical façade outlines in the image and matching them with corresponding model façade outlines, resulting in a sparse set of 3D location hypotheses. To improve the detection of potential vertical façade outlines in the image, we first apply a multi-scale window detector before extracting the dominant vertical lines. We verify the set of pose hypotheses with an objective function that scores the match between a semantic segmentation of the input image and the reprojection of the 2.5D façade model. The semantic segmentation is computed with a fast light-weight multi-class support vector machine.

The resulting global 6DOF keyframe pose together with the retrieved 2.5D model is used by the SLAM system to initialize its 3D map (Section 6). We render a depth map and assign depth values to 2D keyframe features and thus initialize a 3D feature map. This procedure is repeated for subsequent keyframes to extend the 3D map, allowing for absolute 6DOF tracking of arbitrary camera motion in a global coordinate system.

4 Orientation Estimation

We describe the estimation of the absolute orientation of the given camera image with respect to the 2D map. We start by computing the pitch and roll (Section 4.1), *i.e.*, the orientation of the vertical camera axis with respect to gravity, followed by computing the yaw (Section 4.2), *i.e.*, the remaining degree-of-freedom of the rotation in the absolute coordinate system of the 2D map.

4.1 Estimating the vertical axis

We want to estimate a rotation matrix $\mathbf{R}_v$ that aligns the camera's vertical axis with the gravity vector. We do so by determining the dominant vertical vanishing point in the image, using line segments extracted from the image. We rely on the Line Segment Detector (LSD) algorithm [40], followed by three filtering steps: (1) We only retain line segments exceeding a certain length. (2) Lines below the horizon line computed from the sensor rotation estimate are removed, since these segments are likely located on the ground plane or foreground object clutter. (3) Line segments are removed, if the angle between their projection and the gravity vector given by the accelerometer sensor is larger than a threshold [19].

The intersection point $\mathbf{p}$ of the projections $\mathbf{l}_1$ and $\mathbf{l}_2$ of two vertical lines is the vertical vanishing point. It can be computed with as a cross product using homogeneous coordinates:

$$\mathbf{p} = \mathbf{l}_1 \times \mathbf{l}_2 \ . \tag{1}$$

As suggested by Rother *et al*. [29], we search pairs of lines in order to find the dominant vanishing point. For each pair of vertical line segments, we compute the intersection point and test it against all line segments, using an angular error measure:

$$\mathrm{err}(\mathbf{p},\mathbf{l}) = \mathrm{acos}\left(\frac{\mathbf{p}\cdot\mathbf{l}}{||\mathbf{p}||\cdot||\mathbf{l}||}\right) \ . \tag{2}$$

The dominant vertical vanishing point $\mathbf{p}_v$ is chosen as the one with the highest number of inliers using an error threshold of σ degrees, evaluated in a RANSAC framework.

Given the dominant vertical vanishing point $\mathbf{p}_v$, we compute the rotation which aligns the camera's vertical axis with the vertical vanishing point of the 2D map. The vertical direction of the 2D map is assumed $\mathbf{z} = [0\ 0\ 1]^\top$. Besides that, no other information from the 2D map is needed.

[4] UTM: Universal Transverse Mercator

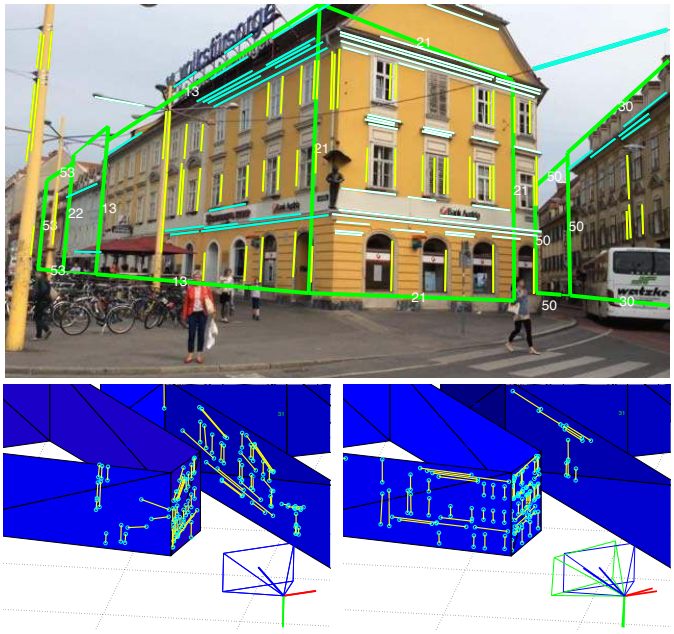

Fig. 3: Estimating rotation $\mathbf{R}$. **Top:** line segments identified as vertical (yellow) and horizontal (cyan, white) in 3D compared to the reprojection (green) of the model after rotation correction, but before translation correction. **Bottom left:** Back-projection of the line segments using the sensor pose. **Bottom right:** Back-projection after rotation correction, which aligns the vertical and horizontal line segments with the façade model.

Using angle-axis representation, the axis of the rotation is $\mathbf{u} = \mathbf{p}_v \times \mathbf{z}$ and the angle is $\theta = \text{acos}(\mathbf{p}_v \cdot \mathbf{z})$, assuming that the vertical vanishing point is normalized. The rotation $\mathbf{R}_v$ then can be constructed using $SO(3)$ exponentiation:

$$\mathbf{R}_v = \exp_{SO(3)}\left(\mathbf{u} \cdot \frac{\theta}{||\mathbf{u}||}\right) . \tag{3}$$

4.2 Orientation in absolute coordinates

At this point, the camera orientation is determined up to a rotation around its vertical axis (yaw). In this section, we explain how to estimate this last degree of freedom for the orientation in the absolute coordinate system of the 2D map. The procedure is illustrated in Fig. 3: The image line segments are rotated and and back-projected onto an extruded 2D model. The optimal rotation $\mathbf{R}$ makes the back-projections as vertical and horizontal as possible. Note that for estimating $\mathbf{R}$, we do not require the building façade heights, but only rely on a 2D map that provides oriented line strips which allow for retrieving the building façade normals.

Given a façade f from the 2D map, its horizontal vanishing point is found as the cross product of its normal $\mathbf{n}_f$ and the vertical axis $\mathbf{z}$:

$$\mathbf{p}_h = \mathbf{n}_f \times \mathbf{z} . \tag{4}$$

After orientation correction through $\mathbf{R}_v$, the horizontal image lines $\mathbf{l}$ lying on f should intersect $\mathbf{p}_h$. Thus, given a horizontal vanishing point $\mathbf{p}_h$ and a rotated horizontal line segment $\mathbf{l}_3 = \mathbf{R}_v^\top \mathbf{l}$, we can compute the rotation $\mathbf{R}_h$ about the vertical axis to align the camera's horizontal axis with the horizontal vanishing point of f. This rotation has one degree of freedom, ϕ_z, the amount of rotation about the vertical axis:

$$\mathbf{R}_h = \begin{bmatrix} \cos\phi_z & -\sin\phi_z & 0 \\ \sin\phi_z & \cos\phi_z & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{5}$$

Using the substitution $q = \tan\frac{\phi_z}{2}$, we get $\cos\phi_z = \frac{1-q^2}{1+q^2}$ and $\sin\phi_z = \frac{2q}{1+q^2}$ [18]. We can parameterize our rotation matrix in terms of q:

$$\mathbf{R}_h = \frac{1}{1+q^2}\begin{bmatrix} 1-q^2 & -2q & 0 \\ 2q & 1-q^2 & 0 \\ 0 & 0 & 1+q^2 \end{bmatrix} . \tag{6}$$

The intersection constraint between $\mathbf{l}_3$ and the horizontal vanishing point $\mathbf{p}_h$ is expressed as

$$\mathbf{p}_h \cdot (\mathbf{R}_h \mathbf{l}_3) = 0 . \tag{7}$$

The roots of this quadratic polynomial in q determine two possible rotations. This ambiguity is resolved by choosing the rotation which best aligns the camera's view vector to the inverse normal $-\mathbf{n}_f$.

Finally, the absolute rotation $\mathbf{R}$ of the camera is computed by chaining the two previous rotations $\mathbf{R}_v$ and $\mathbf{R}_h$:

$$\mathbf{R} = \mathbf{R}_v \mathbf{R}_h . \tag{8}$$

In practice we create pairs $< \mathbf{l}, f >$ from line segments $\mathbf{l}$ assigned to visible façades f, identified from the 2D map using the initial pose estimate from the sensors. We use a Binary Space Partition (BSP) tree for efficient search the 2D map for visible façades – a BSP tree is a data structure from Computer Graphics to efficiently solve visibility problems [12]. We evaluate the angular error measure from Eq. (2) for a rotation estimate from the pair $< \mathbf{l}, f >$ in a RANSAC framework, choosing the hypothesis with the highest number of inliers.

We have to consider the following degenerate case: that is, $< \mathbf{l}, f >$ pairs where $\mathbf{l}$ is actually located on a *perpendicular* façade plane $f_\perp$, resulting in rotation hypotheses $\mathbf{R}$ which are 90 *deg* off the ground truth. Given a visible façade set where all façades are pairwise perpendicular, such a rotation hypothesis may receive the highest number of inliers. We discard such $< \mathbf{l}, f_\perp >$ pairs by computing the angular difference between the sensor pose and the rotation hypothesis $\mathbf{R}$ and discard the hypothesis if it exceeds a threshold of 45 *deg*.

The case of $< \mathbf{l}, f >$ pairs where $\mathbf{l}$ is actually located on a *parallel* façade $f_\parallel$ does not cause any problems because in this case $f_\parallel$ and f have the same horizontal vanishing point $\mathbf{p}_h$.

5 Translation Estimation

The vertical and horizontal segments on the façades allow estimation of the camera's orientation in a global coordinate system. Unfortunately, the segments do not provide any useful constraint to estimate the translation since we do not know their exact 3D location. In theory, the pose could be computed from correspondences between the edges of the buildings in the 2D map and their reprojections in the images. In practice, it is virtually impossible to directly obtain such matches reliably in absence of additional information.

We approach this problem by aligning the 2D map with a semantic segmentation of the image. We can estimate the translation of the camera as the one that aligns the façades of the 2D map with the façades extracted from the image.

To speed up this alignment and to make it more reliable, we first generate a small set of possible translations given the line segments in the image that potentially correspond to the edges of the buildings in the 2D map. We then keep the hypotheses that best aligns the 2D map with the segmentation. We give details on these two steps below.

5.1 Generating translation hypotheses

In practice, the translation along the vertical axis is the most problematic one to estimate from the image, because the bottoms of the buildings are typically occluded by foreground clutter (cars, pedestrians). We therefore simply set the height of the camera at 1.6 *m*, which is reasonable for a handheld device.

For the remaining two degrees of translational freedom, we generate possible horizontal translations (parallel to the ground plane) for the camera by matching the edges of the buildings with the image. However, this is a very challenging task, as the images are very cluttered in practice.

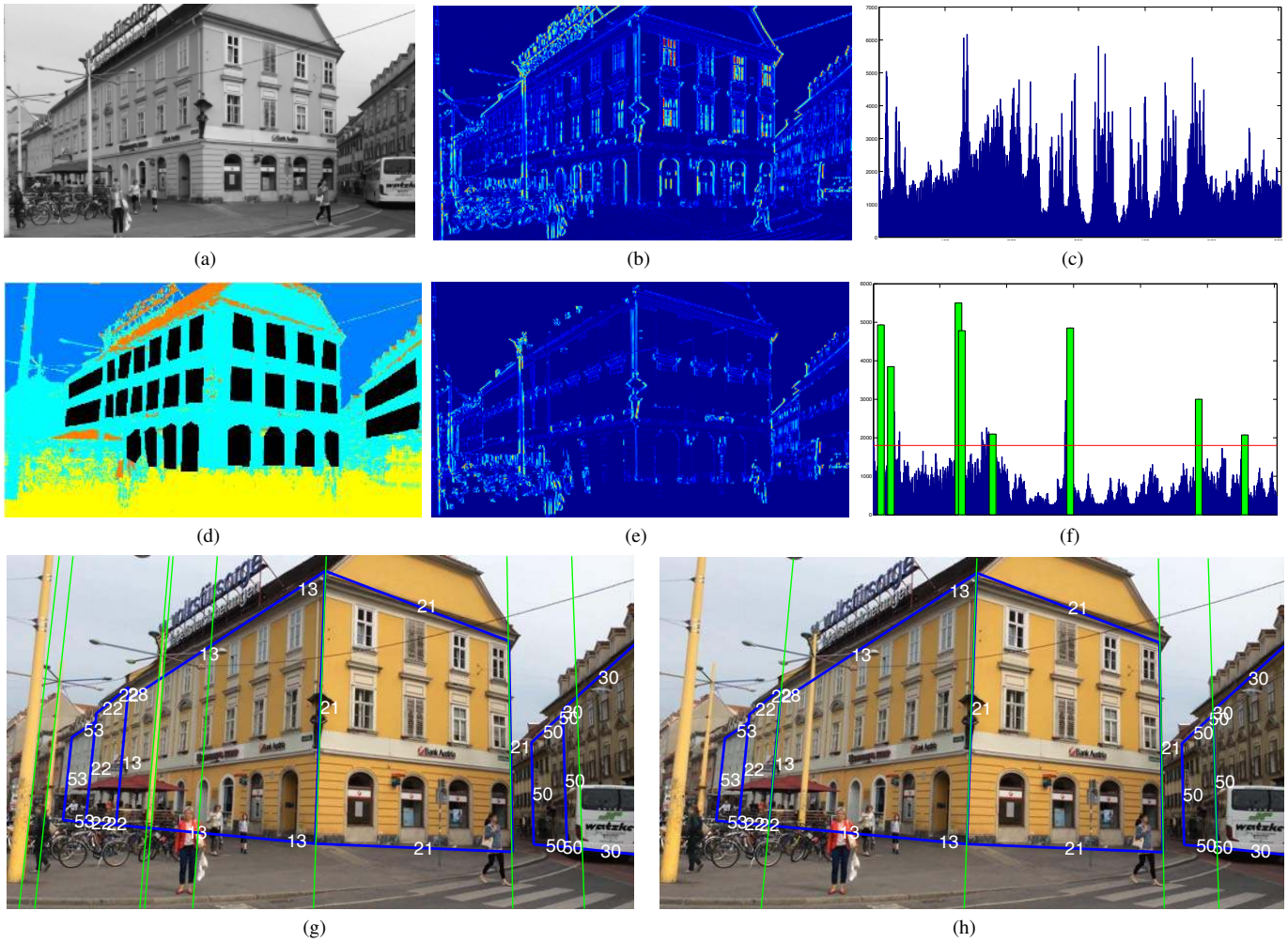

Fig. 4: Generating translation hypotheses. **(a)**: Vertically rectified image. **(b)**: Image gradients. **(c)**: Histogram of the sums of the gradient magnitude over the columns. **(d)**: Segmentation of the façades in cyan and window detections in black. **(e)**: Image gradients only for the pixels lying on a façade, but not on a window. **(f)**: Histogram of gradient sums and selected vertical image lines. **(g)**: Selected image lines overlaid on the original image. **(h)**: 3D model lines from building corners overlaid on the original image using the ground truth pose. Most of the visible building outlines were successfully detected with our method.

Fig. 5: Pixel-wise segmentations obtained with a multi-class SVM for two different images. Cyan corresponds to façades, blue to sky, orange to roofs, green to vegetation, and yellow to ground plane.

Fig. 6: Computing the log-likelihood. **Left:** Probability map for c_f, the façade class. **Middle:** Probability maps for c_s, c_r, c_v and c_g. **Right:** Reprojection $\mathrm{Proj}(M, \mathbf{p})$ for a pose close to the ground truth.

As shown in Fig. 4, we generate a set of possible image locations for the edges of the buildings with a heuristic. We first rectify the input image using the orientation so that vertical 3D lines also appear vertical in the image, and we sum the image gradients along each column.

The columns with a large sum are likely corresponding to the border of a building. However, since windows also have strong vertical edges, they tend to generate many wrong hypotheses. To reduce their influence, we trained a multi-scale window detector based on the work of Viola and Jones [39] on the ZuBud building database [34]. Only almost frontally viewed windows were manually extracted and used for training, resulting in a set of 1170 positive images. As negatives, a set of private images from travels and parties was used. The resulting detector has 28 stages with a total of 651 features. Despite this simple procedure, the detector works reasonably well, still leaving a lot of room for optimization, both in terms of overall cascade depth as well as the selection of negative imagery given the expected urban streetview domain (see *e.g.* [14]).

Pixels lying on the windows found by the detector are ignored when computing the gradient sums over the columns. We also use the façade segmentation result described in Section 5.2 to consider only the pixels that lie on façades, but not on windows. Since the sums may take very different values for different scenes, we use a threshold estimated automatically for each image. We fit a Gamma distribution to the histogram of the sums and evaluate the quantile function with a fixed inlier probability.

Finally, as shown in Fig. 4(g) and Fig. 4(h), we generate translation hypotheses for each possible pair of correspondences between the vertical lines extracted from the image and the building outlines. The building outlines come from the corners in the 2D maps that are likely to be visible, given the location provided by the GPS and the orientation estimated during the first step, again using the BSP tree for efficient retrieval. Given two vertical lines in the image, $\mathbf{l}_1$ and $\mathbf{l}_2$, and two 3D points which are the corresponding building corners, $\mathbf{x}_1$ and $\mathbf{x}_2$, the camera translation $\mathbf{t}$ in the ground plane can be easily computed by solving the following linear system:

$$\begin{cases} \mathbf{l}_1 \cdot (\mathbf{x}_1 + \mathbf{t}) &= 0 \\ \mathbf{l}_2 \cdot (\mathbf{x}_2 + \mathbf{t}) &= 0 \end{cases} . \tag{9}$$

We filter the hypotheses set based on their estimated 3D location. First, hypotheses which have a location outside of a sphere whose radius is determined by the assumed GPS error of 12.5 m [43] are discarded. Second, we remove hypotheses which are located within buildings.

5.2 Aligning the 2.5D map with the image

To select the best translation among the ones generated using the method described above, we evaluate the alignment of the image and the 2.5D map after projection using each generated translation.

We use a simple pixel-wise segmentation of the input image, by applying a classifier to each image patch of a given size to assign a class label to the center location of the patch.

The segmentation uses a multi-class Support Vector Machine (SVM) [33, 6], trained on a dataset of images from a different source than the one used in our evaluations, manually segmenting the images using the *LabelMe* service [30]. We use the integral features introduced by Dollar *et al.* [9], and consider five different classes $C = \{c_f, c_s, c_r, c_v, c_g\}$ for *façade, sky, roof, vegetation* and *ground*, respectively. By applying the classifier exhaustively, we obtain a probability estimate p for each image pixel over these classes. Fig. 5 shows an example of a segmentation for a typical input image.

As illustrated in Fig. 6, given the 2D projection $\mathrm{Proj}(M, \mathbf{p})$ of our 2D map+height M into the image using pose hypothesis $\mathbf{p}$, we compute the log-likelihood of the pose:

$$s_{\mathbf{p}} = \sum_{i}^{\mathrm{Proj}(M,\mathbf{p})} \log p_i(c_f) + \sum_{i}^{\neg\mathrm{Proj}(M,\mathbf{p})} \log\left(1 - p_i(c_f)\right) , \tag{10}$$

where $\neg\mathrm{Proj}(M, \mathbf{p})$ denotes the set of pixels lying outside the reprojection $\mathrm{Proj}(M, \mathbf{p})$. The pixels lying on the projection $\mathrm{Proj}(M, \mathbf{p})$ of the façades should have a high probability to be on a façade in the image, and the pixels lying outside should have a high probability to not be on a façade. We keep the pose $\hat{\mathbf{p}}$ that maximizes the log-likelihood:

$$\hat{\mathbf{p}} = \arg\max_{\mathbf{p}} s_{\mathbf{p}} . \tag{11}$$

In practice, the 3D location estimated from the sensors is often not accurate enough to directly initialize our method. We therefore sample six additional initial locations around the sensor pose in a hexagonal layout, and combine the locations with the previously estimated orientation. We execute our method initialized from each of these seven poses, searching within a sphere having 12.5 m radius [43] for each initial pose. Thus, our method can find the correct image location within a region of up to $40x40\ m$. Finally, we keep the computed pose with the largest likelihood.

Note that this approach naturally extends to more complex building models, for example, if the roofs of the buildings are also present in the model. The log-likelihood then becomes:

$$s_{\mathbf{p}} = \sum_{c \in C_M} \sum_{i}^{\mathrm{Proj}(M_c,\mathbf{p})} \log p_i(c) + \sum_{i}^{\neg\mathrm{Proj}(M,\mathbf{p})} \log\left(1 - \sum_{c \in C_M} p_i(c)\right) , \tag{12}$$

where C_M is a subset of C and made of the different classes that can appear in the buildings model, and $\mathrm{Proj}(M_c, \mathbf{p})$ is the projection of the components of the buildings model for class c.

Much more sophisticated methods could be used [35, 10], but we have empirically verified that the camera translation is reliably computed despite the relatively limited quality of our segmentation.

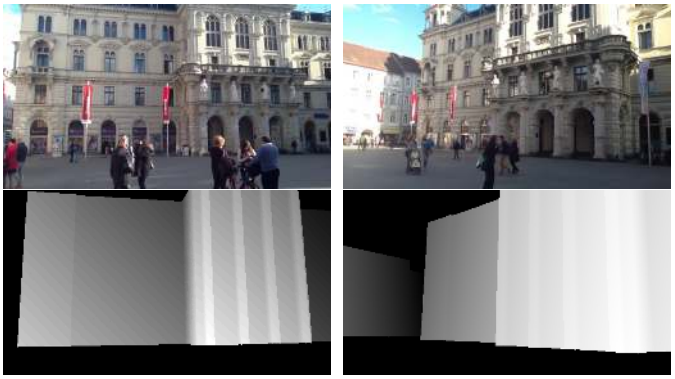

Fig. 7: **Top row:** Two keyframes from the globally aligned SLAM system on a test sequence. **Bottom row:** Corresponding depth images rendered from the actual camera pose.

6 SLAM INITIALIZATION USING DEPTH FROM 2.5D MODELS

Recent SLAM systems for indoor use often rely on depth sensors for superior robustness and instant initialization. These sensors are not available outdoors, but we can use the 2.5D map to generate and use *synthetic* depth images as a cue for mapping and tracking.

We use a keyframe-based SLAM system, similar to PTAM [16]. The tracking and the mapping thread run asynchronously and periodically exchange keyframe and map information. Our localization approach registers the first keyframe to the 2.5D map. Using this pose estimate, we render a polygonal model using the graphics hardware and retrieve the depth buffer to assign depth to those map points, which correspond to observed façades. We arrive at a full 3D map already for the first keyframe. Previous approaches required establishing a baseline of several meters between the first two keyframes for initial triangulation [38].

As the SLAM system acquires more keyframes, the procedure is repeated, and tracked map points collect multiple observations for real triangulation once the baseline between keyframes is sufficient. Fig. 7 shows two keyframes of a test sequence and the corresponding depth images.

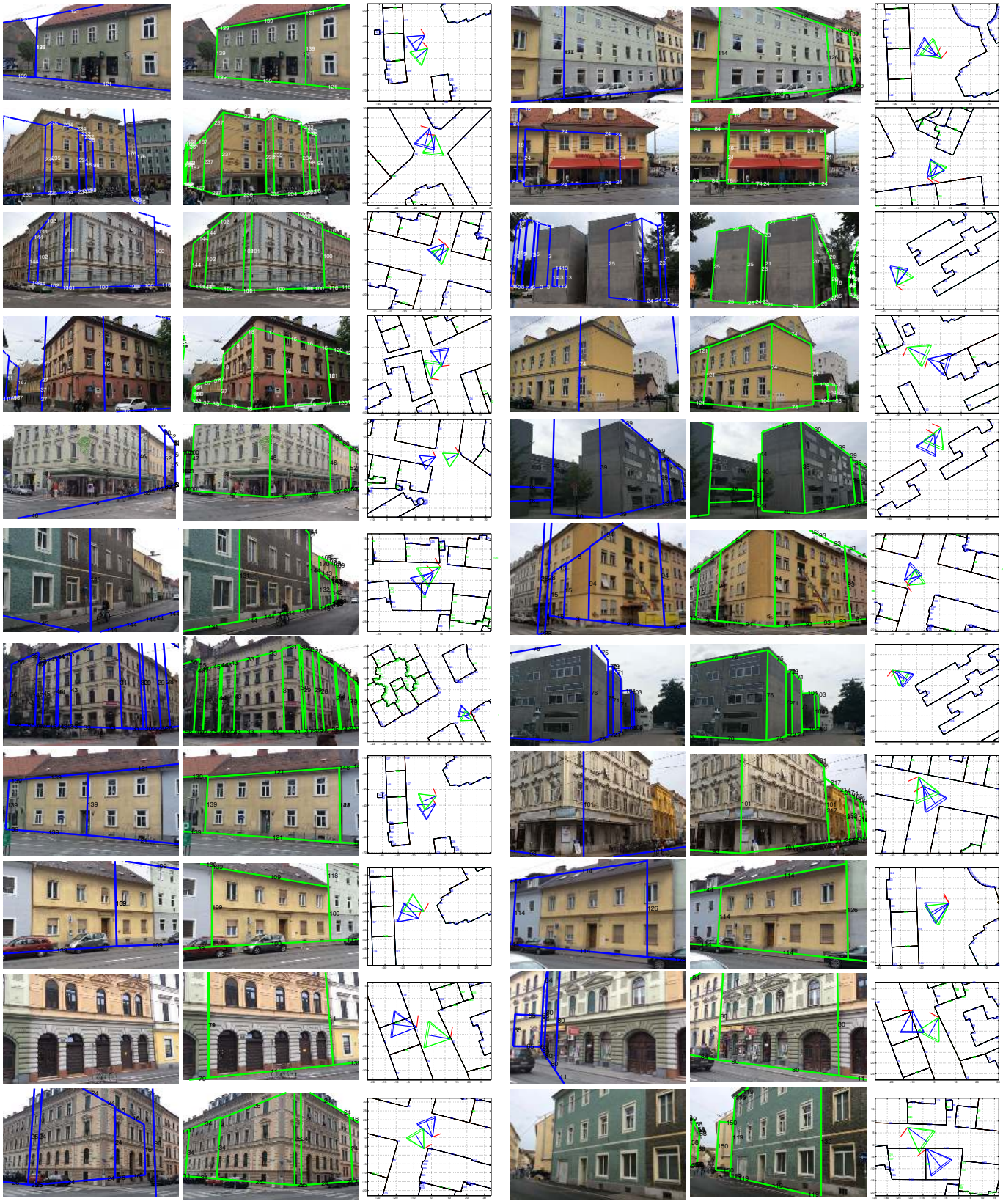

Fig. 8: Results of our approach on test images. For each triplet of images: **Left:** Model reprojection into the image using the initial sensor pose. **Middle:** Model reprojection into the image using the final estimated pose. **Right:** Map view, sensor pose (in blue) and the corrected pose (in green).

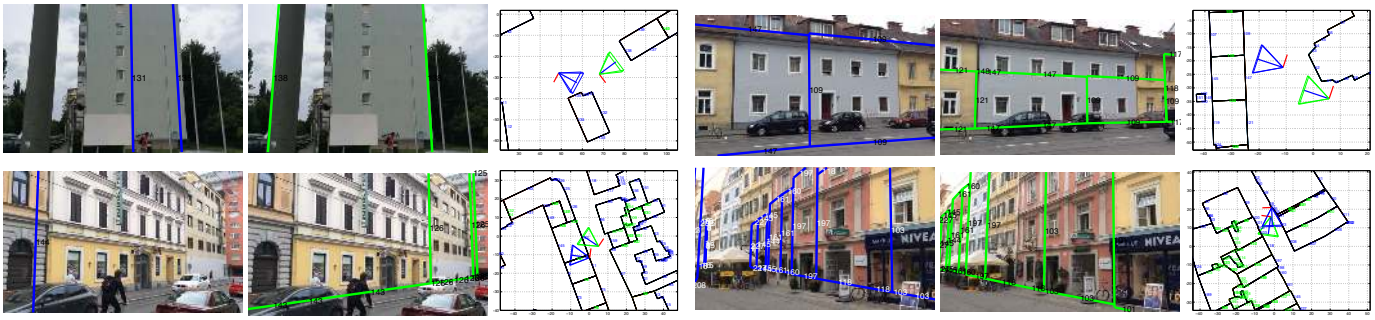

Fig. 9: Images with the largest pose errors. For each triplet of images: **Left:** Model reprojection into the image using the initial sensor pose. **Middle:** Model reprojection into the image using the final estimated pose. **Right:** Map view, sensor pose (in blue) and the corrected pose (in green). Even for these images, the model reprojection tends to be close to the expected position.

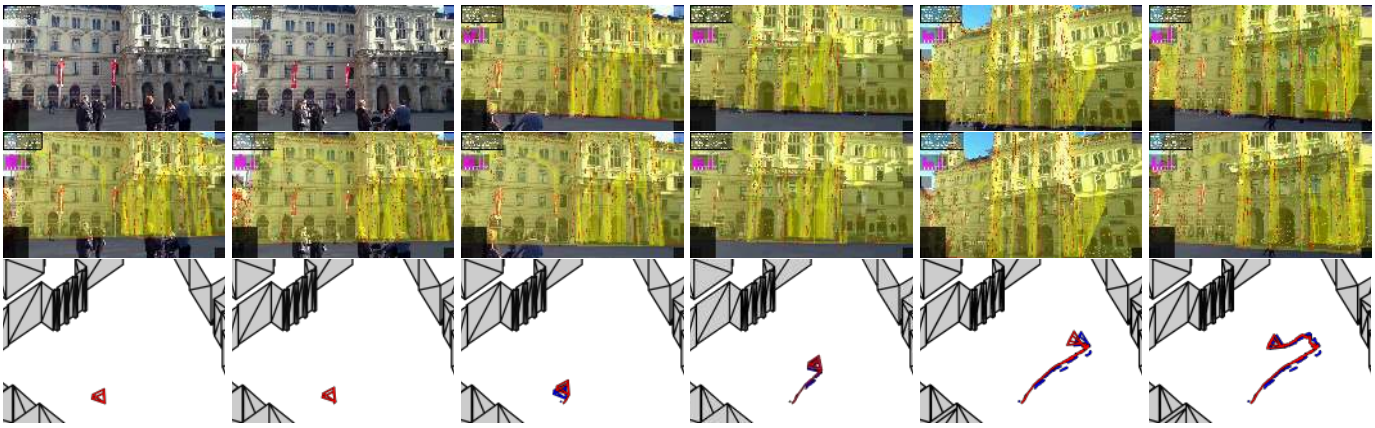

Fig. 10: Comparison of previous work and our approach, together with the estimated camera trajectories. **Top row:** Results from previous work [38]. Due to the required baseline between keyframes, the system initializes after about 12 seconds. **Middle row:** Results from our approach. **Bottom row:** Trajectories estimated by previous work (blue) and our approach (red). Note that the trajectory estimated by our new approach is considerably smoother.

Fig. 11: **Top row:** Annotation results from a rotation-only sequence. **Bottom row:** Expanding SLAM map, while the user rotates a handheld device. The map nicely resembles the structure of the surrounding buildings.

7 Experimental Results

In this section, we first describe the dataset we built to evaluate our localization approach, and then report and discuss the results of the evaluation. Finally, we demonstrate its application to globally-aligned instant urban outdoor SLAM.

7.1 Dataset

To demonstrate the applicability of our approach, we captured a dataset of 32 images with an *Apple iPad Air* in urban and suburban environments of Graz, Austria [5].

[5]We will make our dataset publicly available, including 32 images, sensor and ground truth poses, 2D+height maps.

The images were captured without any special consideration for satellite shadowing or surrounding metallic structures. As a consequence, the accuracy of the pose estimated with the sensors only ranges from very accurate, about 0.4 *m* position and 2° rotation error, to very poor, up to 16.5 *m* position and about 30° rotation error. Since altitude estimates from sensors tend to be very poor, we overrode these estimates of the poses predicted by the sensors with a default value of 1.6 *m*. For each test image, we calculated a ground truth pose by manually matching 2D image locations with 3D points from the maps.

We retrieved 2D maps of the surroundings from OpenStreetMap and extruded them with a coarse estimate of the height of the building façades. OpenStreetMap data consists of oriented line strips, which we converted into a triangle mesh including face normals. Each building

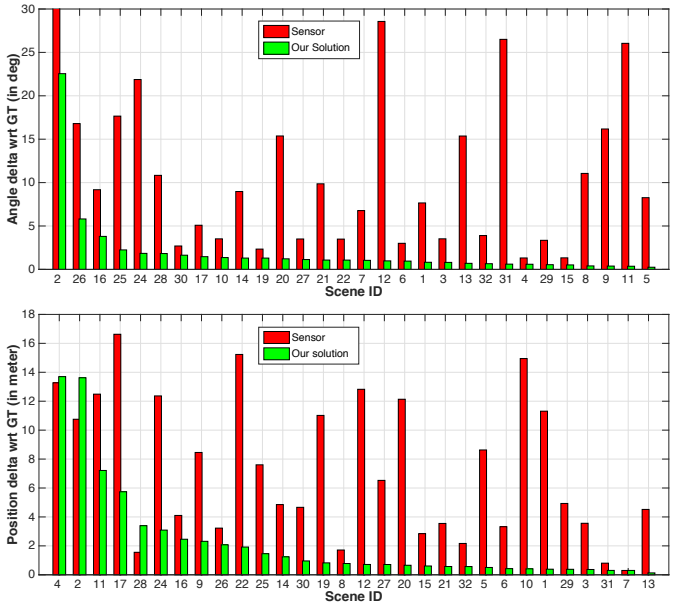

Fig. 12: Pose estimates accuracy. **Top:** Orientation, and **Bottom:** Translation. We ranked the images from the one with the largest error after correction to the one with the smallest error. Our method significantly improves the accuracy of the orientation and translation estimates.

façade plane is modeled as 2D quad with four vertices, two ground plane vertices and two roof vertices. The heights of the vertices were taken from aerial laser scan data. All vertical building outlines were aligned with the global vertical up-vector.

7.2 Orientation and Translation accuracy

Fig. 12, top, plots the angular error of the camera pose predicted by the sensors and after correction with our method. The error is calculated as the angular difference between the estimated rotation and the ground-truth rotation in angle-axis representation. We ranked the images from the one with the largest error after correction to the one with the smallest error. The sensor error can become as large as 30°. With our method, all our orientation estimates have an angle error below 5°, with the exception of a single outlier image which contains very few horizontal lines. 90.6% of the estimates are below 3°, 84.4% below 2° and 50% below 1° of angular error w.r.t. the ground truth rotation.

Fig. 12, bottom, gives the results of our translation estimation method. As for the rotation, we ranked the images from the one with the largest error after correction to the one with the smallest error. The sensor errors range from about 0.4 m to about 16.5 m, with an average error of about 8 m. Our method significantly decreases the translation error in most of the cases. The worst results are due to adjacent buildings, with edges that cannot be extracted correctly. Overall, our method is able to considerably improve the position estimates from the sensors, with the pose estimates for 87.5% of the images being below 4 m, 68.8% below 2 m and 59.4% below 1 m of error w.r.t. the ground truth position.

7.3 Visual inspection

Fig. 8 presents the final results of our algorithm for a wide variety of test images, showing the reprojection of the model using both the sensor pose and the pose retrieved from our approach. After pose estimation, the outlines of the models nicely fit the building outlines, even for very challenging scenes with many façades visible and a considerable rotation and position error in the sensor estimate. The amount of correction can be assessed from the map view, as both the rotation and translation undergo a significant correction during the application of our method.

Fig. 9 shows the images with the largest pose errors. In the upper left scene the algorithm is fooled by a street lamp, in the upper right scenario the classification result is bad around the window areas, finally resulting in a wrong estimate of the scene distance. In the lower left scenario, a model of the background building is missing. In the lower right case, the classification result and the model line selection is bad, causing the translation estimation to fail.

7.4 Unconstrained SLAM using depth images

We evaluated the integration of the single image geo-localization for initializing our SLAM system on multiple sequences, featuring both rotation-only and general camera motion.

Our new approach has noticeable benefits over conventional SLAM in urban outdoor environments, as it provides accurate 6DOF localization right from the start. Usability is considerably improved, in particular for panoramic camera motion, which users can hardly avoid [13, 24]. A direct comparison to Ventura *et al*. [38] with frames of a SLAM sequence is given in Fig. 10. In the previous approach, it takes more than 12 seconds for the system to successfully initialize, while our approach provides a globally accurate 6DOF pose for SLAM starting from the first captured image. Note that the estimated camera trajectory is considerably smoother, because the 3D locations of feature points are constrained by projecting them onto the façades.

Fig. 11 shows results of augmentations for a rotation-only sequence together with the evolving SLAM map over time as the user rotates the device. Note that the planar labels nicely align with the real building façades, and the recovered 3D SLAM maps resemble the surrounding building structure quite accurately.

8 Discussion

In the following, we critically discuss our results with respect to the current status of our framework and potential improvements.

8.1 Localization method

Our localization method gives good results on a large number of diverse scenes. The method is very versatile since it only sets one strict requirement onto an input image: the visibility of two vertical building façade outlines – meaning that a façade may also be truncated towards the sky and/or towards the ground plane. Furthermore, we can cope with considerable sensor pose errors, potentially as much as 45deg rotation offset and up to 40 m of position offset.

We identified the main reasons for a complete failure of our method to be cases where the pose prior from sensors is unusable – either, because it is inside a building or because it is misplaced in an a incorrect street segment, where the correct building façades cannot be observed. A simple improvement for the former case is to move the pose prior to the nearest street location, but this heuristic is not guaranteed to solve the problem in all cases.

The main reason for errors in our orientation estimation is an insufficient number of suitable vertical or horizontal lines found in an image to allow for robust vanishing point estimation. Tweaking the line detection parameters might partially overcome this problem for certain scenes.

Errors in our translation estimation are mainly due to the inability of our algorithm to detect the correct façade outlines for various reasons, *e.g*. because the orientation estimate is bad, adjacent façades look too similar, the façade texture contains too much clutter or the window detector does not properly filter out windows. Similarly, the scoring function might select the wrong hypothesis, because of a bad semantic segmentation result or a bad reprojection result owed to insufficient model detail. All individual steps might undergo additional tuning to improve the overall performance of the approach.

Currently, the pose computation is implemented as unoptimized, single-threaded Matlab code and requires about 30 seconds per frame on a single CPU core on a 2011 Intel 2.5 GHz i5 Macbook Pro for a 640×360 image. The execution time for the individual major parts of the algorithm are given in Table 1. However, each of these steps have been demonstrated at much higher speed with optimized code and a GPU. For example, most of the time for the window detection step is used for image warping. Using the GPU would provide a significant speedup.

Part	Algorithm	Approx. Time [s]
(i)	Window detection	10
(ii)	Segmentation	14
(iii)	Translation estimation	6

Table 1: Timings of major parts of our unoptimized localization procedure running mainly in Matlab on a single CPU core on a 2011 Intel 2.5 GHz i5 Macbook Pro for a 640×360 image

8.2 SLAM

The SLAM system is very robust, runs in real-time on current mobile devices and works sufficiently well even under difficult lighting conditions outdoors. Currently, the system uses a single localization result to initialize the 3D map and to maintain it over time through rendering new keyframes and extending the map with new 3D points. However, even a small rotational error (< 1 degree) or a position error below $25cm$ accumulated through drift in the system over time might create a noticeable offset in the visualization of annotations in the distance. Therefore, the use of multiple localization results over time and the inclusion of more advanced methods in the iterative optimization of the system (*i.e*. bundle adjustment) would improve the overall system robustness and accuracy considerably, and is a topic of future work.

8.3 AR content and models

Our entire approach is designed to take only the minimum information into account one might hope for to be available anywhere globally: a 2D map and some building height information. Naturally, with more detailed and accurate models, even better results could be achieved. Also note Eq. (12), where we mentioned the natural extension of our approach in terms of semantic information in the segmentation stage.

In AR, this raises an important point where synergies can be exploited: as there needs to be annotated content to be visualized, this information can in turn feedback into the localization approach to improve localization performance. For example, using the AR annotations of windows or doors can be used in connection to our window detector to add another semantic class to the scoring function. We therefore argue that certain AR content might itself be used to improve localization performance within our framework, although this content is largely missing at this point in time.

9 Conclusion

We have presented a novel approach for accurate global 6DOF pose localization and tracking that relies only on components readily available for urban outdoor AR: mobile sensors, narrow field-of-view images and open-source 2.5D maps provide a pose exploitable by SLAM applications for initialization. Our approach therefore offers new opportunities for making AR practical in outdoor urban environments.

References

[1] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide Area Loc. on Mobile Phones. In *ISMAR*, pages 73–82, 2009.

[2] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys. Large Scale Visual Geo-Localization of Images in Mountainous Terrain. In *ECCV*, 2012.

[3] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys. Leveraging Topographic Maps for Image to Terrain Alignment. *3DPVT*, pages 487–492, 2012.

[4] M. Bansal and K. Daniilidis. Geometric Urban Geo-Localization. In *CVPR*, 2014.

[5] T. J. Cham, A. Ciptadi, W. C. Tan, M. T. Pham, and L. T. Chia. Estimating camera pose from a single urban ground-view omnidirectional image and a 2D building outline map. *CVPR*, pages 366–373, 2010.

[6] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, 2011.

[7] H. Chu, A. Gallagher, and T. Chen. GPS Refinement and Camera Orientation Estimation from a Single Image and a 2D Map. In *CVPR*, 2014.

[8] P. David and S. Ho. Orientation descriptors for localization in urban environments. *IROS*, pages 494–501, 2011.

[9] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009.

[10] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning Hierarchical Features for Scene Labeling. *PAMI*, 2013.

[11] S. Feiner, B. MacIntyre, T. Höllerer, and A. Webster. A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment. *Pers. and Ubiquitous Comp.*, 1(4):208–217, 1997.

[12] H. Fuchs, G. D. Abram, and E. D. Grant. Near real-time shaded display of rigid objects. *SIGGRAPH Comput. Graph.*, 17(3):65–72, July 1983.

[13] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Hollerer. Live track. and mapp. from both gen. and rot.-only camera motion. In *ISMAR*, 2012.

[14] V. Jain and E. Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. In *CVPR*, pages 577–584, June 2011.

[15] B. Jiang, U. Neumann, and S. You. A Robust Hybrid Tracking System for Outdoor Augmented Reality. In *VR*, 2004.

[16] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR*, November 2007.

[17] J. Krolewski and P. Gawrysiak. The Mobile Personal Augmented Reality Navigation System. In *Man-Machine Interactions 2*, volume 103, pages 105–113. Springer Berlin Heidelberg, 2011.

[18] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-Form Solutions to Min. Absolute Pose Problems with Known Vertical Direction. In *ACCV*, 2011.

[19] D. Kurz and S. Benhimane. Gravity-Aware Handheld Augmented Reality. In *ISMAR*, 2010.

[20] A. Mastin, J. Kepner, and J. Fisher. Automatic registration of LIDAR and optical images of urban scenes. In *CVPR WS*, pages 2639–2646, 2009.

[21] B. C. Matei, N. Vander Valk, Z. Zhu, H. Cheng, and H. S. Sawhney. Image to LIDAR matching for geotagging in urban environments. In *Proc. of IEEE Workshop on App. of Comp. Vision*, pages 413–420, 2013.

[22] N. Meierhold and A. Schmich. Referencing of images to laser scanner data using lin. feat. extracted from digital images and range images. *Int. Arch. of Photogrammetry and Remote Sensing*, XXXVIII:164–170, 2009.

[23] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt. Scalable 6-DOF Localization on Mobile Devices. In *ECCV*, 2014.

[24] C. Pirchheim, D. Schmalstieg, and G. Reitmayr. Handling pure camera rotation in keyframe-based slam. In *ISMAR*, 2013.

[25] S. Ramalingam, S. Bouaziz, and P. Sturm. Pose Estimation Using Both Points and Lines for Geo-Localization. In *ICRA*, 2011.

[26] G. Reitmayr and T. W. Drummond. Going Out: Robust Model-Based Tracking for Outdoor AR. In *ISMAR*, 2006.

[27] G. Reitmayr and T. W. Drummond. Initialisation for visual tracking in urban environments. In *ISMAR*, 2007.

[28] D. Robertson and R. Cipolla. An Image-Based System for Urban Navigation. In *BMVC*, pages 1–10, 2004.

[29] C. Rother. A New Approach to Vanishing Point Detection in Architectural Environments. *Image and Vision Comp.*, 20(9-10):647–655, 2002.

[30] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: A datab. and web-based tool for image ann. *IJCV*, 77(1-3):157–173, 2008.

[31] T. Sattler, B. Leibe, and L. Kobbelt. Improving Image-Based Localization by Active Correspondence Search. In *ECCV*, 2012.

[32] G. Schindler, M. Brown, and R. Szeliski. City-Scale Location Recognition. In *CVPR*, 2007.

[33] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[34] H. Shao, T. Svoboda, and L. V. Gool. Zubud zurich buildings database for image based recognition. Technical Report 260, 2003.

[35] J. Shotton, M. Johnson, and R. Cipolla. Semantic Texton Forests for Image Categorization and Segmentation. In *CVPR*, 2008.

[36] A. Taneja, L. Ballan, and M. Pollefeys. Reg. of Spherical Panoramic Im. with Cadastral 3D Models. *3DPVT*, pages 479–486, Oct. 2012.

[37] G. Vaca-Castano, A. R. Zamir, and M. Shah. City Scale Geo-Spatial Trajectory Estimation of a Moving Camera. In *CVPR*, 2012.

[38] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. Global Localization from Monocular SLAM on a Mobile Phone. In *VR*, 2014.

[39] P. Viola and M. Jones. Robust Real-Time Face Detection. *IJCV*, 57(2):137–154, 2004.

[40] R. von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall. LSD: A Fast Line Segm. Det. with a False Det. Control. *PAMI*, 32(4):722–732, 2010.

[41] K. Xu, A. Cheok, K. W. Chia, and S. Prince. Visual Reg. for Geographical Labeling in Wearable Comp. In *Int. Symp. on Wearable Comp.*, 2002.

[42] A. R. Zamir and M. Shah. Accurate Image Localization Based on Google Maps Street View. In *ECCV*, 2010.

[43] P. A. Zandbergen and S. J. Barbeau. Positional Accuracy of Assisted GPS Data from High-Sensitivity GPS-enabled Mobile Phones. *Journal of Navigation*, 64:381–399, 7 2011.