Learning to Align Semantic Segmentation and 2.5D Maps for Geolocalization

Anil Armagan, Martin Hirzer, Peter M. Roth and Vincent Lepetit
Institute of Computer Graphics and Vision
Graz University of Technology, Austria
{armagan,hirzer,pmroth,lepetit}@icg.tugraz.at

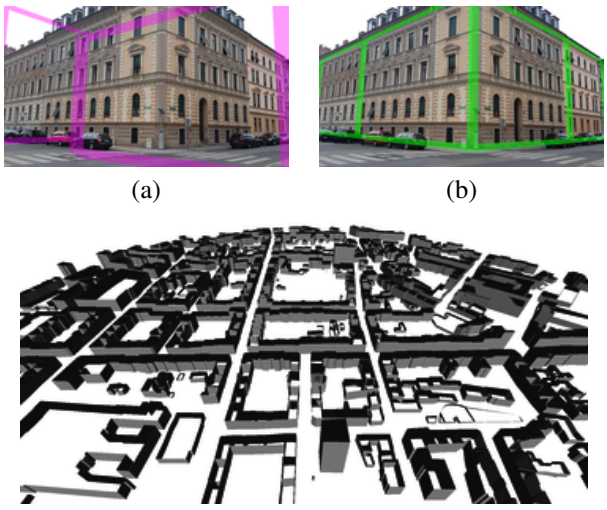
Abstract

We present an efficient method for geolocalization in urban environments starting from a coarse estimate of the location provided by a GPS and using a simple untextured 2.5D model of the surrounding buildings. Our key contribution is a novel efficient and robust method to optimize the pose: We train a Deep Network to predict the best direction to improve a pose estimate, given a semantic segmentation of the input image and a rendering of the buildings from this estimate. We then iteratively apply this CNN until converging to a good pose. This approach avoids the use of reference images of the surroundings, which are difficult to acquire and match, while 2.5D models are broadly available. We can therefore apply it to places unseen during training.

1. Introduction

As recent challenges and benchmarks such as [8, 21] show, dealing with urban scenarios is of increasing interest, including important applications such as autonomous driving and Augmented Reality. One of the key problems for these tasks is the accurate geolocalization of images, which is not easy to solve in practice. Even though sufficient for navigation, GPS information is not accurate enough for many other tasks, especially, if the exact camera pose should be estimated.

Thus, typically image-based localization techniques including for example [22, 23] were introduced. However, they rely on pre-registered images of the surrounding. Therefore, we are facing two problems. First, a large amount of images needs to be captured and registered, which is very cumbersome. Even large collections such as Google Street View[1] are rather sparsely sampled. Second, such data only reflects a very specific appearance of the scene, making a robust matching rather hard under different illumination conditions [30].



(a) (b) (c)

Figure 1. Overview of our approach. We can correct the initial sensor pose (a) to an accurate estimate (b) in locations unseen at training time as long as a 2.5D map (c) of the neighborhood is available. In other words, we learn to align the map with the input image given a coarse estimate of the pose provided by the sensors.

To overcome these drawbacks, we propose a method that, starting from a crude initialization provided by a GPS and orientation sensors, estimates the camera pose from a perspective, non-panoramic input image given 2.5D maps [2] only. This is illustrated in Fig. 1, where we show an example for the initial sensor pose and the corresponding pose recovered with our method along with the used 2.5D map.

In contrast to pre-registered street view data, 2.5D maps are broadly available and easy to obtain, and have been already considered for localization (*e.g.*, [1]). However, as these maps are reduced to the outline and the height of the buildings, our task is getting more difficult. In fact, there is no texture available, which could be used for matching. Therefore, we adopt recent advances in semantic segmentation based on Deep Learning [3,14,20] to extract the façades

[1] https://maps.google.com/help/maps/streetview

[2] In particular, we build on https://www.openstreetmap.org.

and the edges of buildings in the input images. We also use 3D renderings of this map, changing the input image and the 2.5D map into representations that are easier to compare.

However, robustly aligning the input image and the map remains challenging: The initial pose from the sensors can be far away from the correct pose, and direct comparisons between the semantic segmentation of the image and the 3D rendering is getting difficult. Therefore, our contribution is a robust method to optimize the camera pose and to estimate a good alignment between the image and the corresponding map. As gradient methods cannot be applied, using the semantic segmentation and a rendering of the map as input, we train two deep networks – one for the translation and one for the rotation – predicting directions that will improve the current estimate for the pose. By invoking these networks multiple times, we can iteratively improve the estimate of the pose. However, if the initial error is very large, it is not possible to predict a reliable direction. We then run our algorithm from poses sampled around the sensor pose and select the best one. The efficiency and robustness of our approach is demonstrated on a complex real world scenario, where even though starting from a sub-optimal initializations, finally, the correct poses can be estimated.

The remainder of the paper is organized as follows. First, in Section 2, we discuss the related work. Next, in Section 3, we detail our approach and discuss the segmentation step, the prediction models as well as the pose estimation algorithm. Experimental results demonstrating the benefits of the approach are given in Section 4. Finally, we give a conclusion and an outlook in Section 5.

2. Related Work

The main goal of our work is to estimate an accurate and robust localization from a mobile device. A plausible way for doing so would be to use the available GPS position [10], which, however, is often not sufficient to finally obtain accurate results. Thus, given one or more input images and optionally a sensor prior (*e.g.*, GPS or compass information), existing approaches use similar, pre-registered images from a database to compute the pose of the input image. For instance, [23] demonstrates image-based localization using databases that contain 20km of urban street-side imagery, organized in a vocabulary tree to handle the massive amount of data. Later works such as [22] improve upon both, accuracy and performance. Very recently, [13] uses a CNN to predict a 6 DoF camera pose directly from an image, where the idea of transfer learning – from large scale data classification to the task of re-localization – is adopted. However, the underlying network has to be re-trained for each new scene, typically limiting the approach to a certain restricted area that was used for training.

In general, all of these image-based localization approaches do not scale very well. Indeed, 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 or construction activity. To avoid these problems and to avoid the time consuming generation of scene databases, [31] and [27] build their approaches on publicly available existing image collections such as Google Street View and Microsoft StreetSide. However, also these databases are not universally suited for localization, as they are only sparsely sampled and are not available for certain regions and countries.

From a practical point of view, we want to totally avoid the creation of databases on our own and to be more flexible, that is, not being limited by sparsely sampled areas. One way to overcome these problems is to use 2.5D maps (*i.e.*, untextured 2D cadastral maps augmented with height information). For instance, [18] registers an image with respect to a 2.5D model by matching 3D and 2D lines and points. However, in this case a second image, which has already to be registered, is required to establish the 3D-2D correspondences. Consequently, the first image of a sequence needs to be manually annotated. Similarly, [16] establishes line correspondences between the input image and a 2.5D map of the scene. However, due to insufficient accuracy regarding the image orientation, additionally, some kind of user interaction is required. In contrast, our method is not only fully automatic, but also requires only a single input image.

A different approach is to register panoramic images with 2D maps, where the large field-of-view information significantly improves the localization capabilities. For instance, [9] proposes a façade orientation descriptor, however, since mobile devices typically have a rather narrow field of view, such a descriptor is often not discriminative enough. In contrast, [5] aims at detecting vertical building outlines and façade normals, finally yielding 2D fragments, which are then matched against a 2D map. Similarly, [6] matches a descriptor computed from vertical building outlines in perspective input images with a 2D map. As a drawback, however, partially manual input is required to ease the detection of vertical edges and vanishing points.

[17, 29] also consider the buildings' edges, however, [17] relies on orthographic aerial images, which makes the task easier, and [29] assumes that the façades are highly repetitive. [12] uses Auto-context for façades segmentation, where the method is, however, applied to very repetitive façades. In addition, grammar methods are used and complex, handcrafted features are introduced. By contrast, we take advantage of recent advances in semantic segmentation based on CNNs to identify the buildings' edges. In contrast to most of these approaches, our method is fully automatic.

As we do, [24] adopts ideas from segmentation. To find the 3D pose, the façades in the input image are segmented and aligned to a 2.5D map, requiring an optimization in the 6D pose space. In addition, the approach relies on a very detailed 2.5D model and high resolution panoramas from Google Street View in order to provide an accurate initial geolocation. In contrast, [2] registers semantically labeled images with respect to labeled 3D digital terrain models, however, restricting them to simple classes (*i.e.*, water, settlement, other). Moreover, the approach allows only for computing the orientation of the image with respect to the model, but not for estimating its 3D position.

Considering a slightly different problem, semantic information has also been used in the field of 3D scene reconstruction [4, 28], allowing for an interaction between 3D depth estimation and semantic segmentation. In particular, [28] fuses depth images from stereo pairs with a common 3D map using visual odometry, and [4] proposes a large-scale 3D scene reconstruction approach by jointly reasoning about 3D shape and semantic labels. In contrast to these approaches, we show that adopting semantic segmentation to extract the edges of buildings can avoid most of these restrictions and that our method computes both, the absolute orientation and the 3D location.

Different approaches also consider parsing of façades [7, 11, 15, 19, 25], using procedural shape priors and grammars. However, they mostly focus on frontal views of a single façade. By contrast, we consider general images, and our final goal is also very different as we aim at the geolocalization of the camera.

3. Geolocalization with 2.5D Maps

Let be given an input image I_{input} and corresponding sensor information as well as a 2.5D map $\mathcal{M}$ (describing the buildings' outlines and their heights) of the surrounding. Then our goal is to geolocalize a camera in an urban outdoor scene, starting from the initially provided pose estimate $\tilde{\mathbf{p}}$.

In particular, we generate an intermediate representation for I_{input} based on semantic segmentation to link it with the given 2.5D map (see Sec. 3.1). This information is then used in order to train two deep networks to improve the current pose estimate. The first network predicts a direction in space and the second one a direction in orientation (see Sec. 3.2). Finally, these networks are used to estimate the final pose $\mathbf{p}$ (see Sec. 3.3).

3.1. Semantic Segmentation

We use a fully convolutional network (FCN) [14] to segment I_{input} into c semantic classes. We only consider classes that are relevant to our problem and that correspond to elements of the 2.5D map. More exactly, we extract the façades, the vertical and horizontal edges of the façades and a background class (*i.e.*, the sky and the ground plane). In particular, edges are usually not considered in typical semantic segmentation problems, however, they will be useful in our case to disambiguate the pose when, for example, the façades of the buildings are aligned.

3.2. Learning to Predict a Direction

The initial sensor pose $\tilde{\mathbf{p}}$ gives us a coarse estimate of the pose. In practice, the angles with respect to the gravity are well defined via the sensors, giving us two angles of the camera orientation, namely the roll and pitch. As we are using a handheld device, we can also assume a fixed camera height (we use 1.6 m in practice). Thus, only three degrees-of-freedom (along the ground plane) are remaining, two for the location and one for the orientation. However, as these estimates can be very far away from the ground truth, correcting them is challenging.

To deal with this problem, we train two networks to predict directions to improve the pose estimates. The first network predicts a direction for the location. We initially tried to predict a 2D vector pointing to the correct location. This, however, did not succeed, as this problem was too difficult to learn. In fact, the length of the vector would depend on the distances to the buildings, which are lost at least to some extent because of the perspective projection.

Instead, we relax the task and solve a simpler classification problem: We discretize the directions along the ground plane into 8 possible directions, defined in the camera coordinate system. Then, given the semantic segmentation of the image and a rendering of the 2.5D map from the current estimate, we train a network CNN_t to predict the direction that improves the estimated location. We also add a class indicating that the location is already correct and should not be changed. The network CNN_t thus returns a 9-dimensional vector:

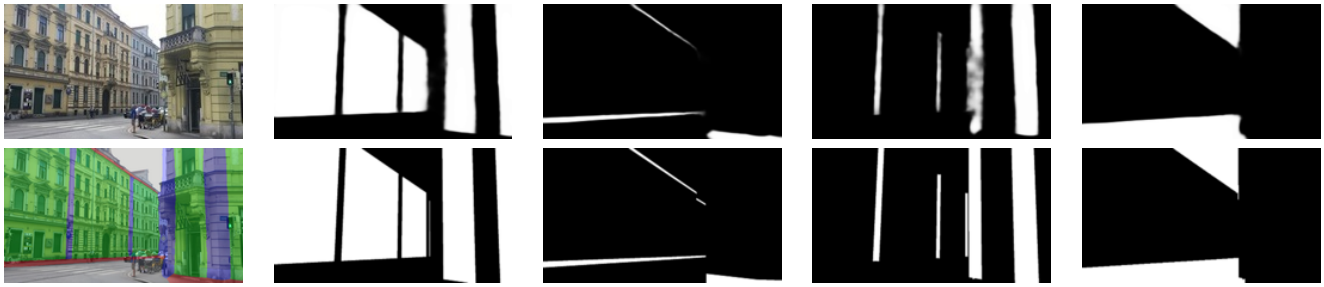
$$\mathbf{d}_t = \text{CNN}_t(R_{\text{F}}, R_{\text{HE}}, R_{\text{VE}}, R_{\text{BG}}, S_{\text{F}}, S_{\text{HE}}, S_{\text{VE}}, S_{\text{BG}}) \,, \quad (1)$$

where S_{F}, S_{HE}, S_{VE}, S_{BG} are the probability maps computed by the semantic segmentation for the input image I_{input} for the classes façade, horizontal edge, vertical edge, and background, respectively. R_{F}, R_{HE}, R_{VE}, R_{BG} are binary maps for the same classes, created by rendering the 2.5D map for the current pose estimate. Examples of these probability and binary maps are shown in Fig. 2. The direction corresponding to the largest value in the output $\mathbf{d}_t$ is the direction predicted by the network.

In addition, we train a second network CNN_o to estimate an update for the orientation:

$$\mathbf{d}_o = \text{CNN}_o(R_{\text{F}}, R_{\text{HE}}, R_{\text{VE}}, R_{\text{BG}}, S_{\text{F}}, S_{\text{HE}}, S_{\text{VE}}, S_{\text{BG}}), \quad (2)$$

where the three values of $\mathbf{d}_o$ indicate if it is best to rotate the camera to the right, to the left, or do not rotate at all.



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

Figure 2. Illustrative example of the inputs to our localization networks: (a) Input image (top) and its segmentation (bottom). (b)–(e): Probability maps S_{F}, S_{HE}, S_{VE}, S_{BG} (top row) and the binary masks R_{F}, R_{HE}, R_{VE}, R_{BG} (bottom row).

We use the same architecture for both networks, CNN_t and CNN_o. Each pair made of a probability map and a rendering for a class is fed to the network along a separate stream. Each stream consists of 2 convolutional layers with 64 and 128 filters, respectively. The sizes of the filters are 5×5 and 3×3. The outputs from the streams are concatenated and fed to fully connected layers: We use three fully connected layers with 1024, 512 and 128 units. The last layer implements a linear logistic regressor. We optimize both networks using the RMSprob [26] algorithm.

How these two networks are applied for pose estimation is described in Sec. 3.3 in more detail. Applying two networks solving separated problems has two main advantages: (1) We do not need to balance between translation and orientation. (2) The resulting optimization problem is easier and can thus be also solved on computationally less powerful devices.

3.3. Pose Estimation Algorithm

Starting from the initial estimate $\tilde{\mathbf{p}}$, we iteratively apply CNN_t and CNN_o and update the current pose after each iteration. In practice, $\tilde{\mathbf{p}}$ can be far away from the correct pose. Moreover, the networks CNN_t and CNN_o introduced above are able to predict a good direction in practice, but do not provide a magnitude.

We therefore use a line search strategy to decide the magnitude of the update. To evaluate the quality of a pose as in [1], we use the maximum log-likelihood:

$$s_{\mathbf{p}} = \sum_{c \in \{\text{F,HE,VE,BG}\}} \sum_{i \in R_c} \log S_c^i \,, \tag{3}$$

where S_c^i is the probability at location i for class c from the semantic segmentation, and $\{i \in R_c\}$ is the set of locations that are set to 1 in the rendered binary mask R_c.

Given one direction by one of the two networks, we evaluate several poses along this direction, and keep the one that maximizes the log-likelihood in Eq. (3). We then switch to the other network. These steps are iterated, and we stop when the two networks predict not to move any more. The overall procedure is summarized in Alg. 1.

Algorithm 1

procedure OPTIMIZEPOSE(I_{input}, $\tilde{\mathbf{p}}$, $\mathcal{M}$)
  $S = (S_{\text{F}}, S_{\text{VE}}, S_{\text{HE}}, S_{\text{BG}}) \leftarrow FCN(I_{\text{input}})$
  $\mathbf{p} \leftarrow \tilde{\mathbf{p}}$
  repeat
    $R = (R_{\text{F}}, R_{\text{VE}}, R_{\text{HE}}, R_{\text{BG}}) \leftarrow \text{render}(\mathbf{p}, \mathcal{M})$
    $d_t \leftarrow \arg\max_i \text{CNN}_t(S, R)[i]$
    if $d_t \neq$ 'do not move' **then**
      $\mathbf{p} \leftarrow \text{lineSearch}_t(\mathbf{p}, d_t, S, \mathcal{M})$
    end if
    $d_o \leftarrow \arg\max_i \text{CNN}_o(S, R)[i]$
    if $d_o \neq$ 'do not move' **then**
      $\mathbf{p} \leftarrow \text{lineSearch}_o(\mathbf{p}, d_o, S, \mathcal{M})$
    end if
  until $d_t =$ 'do not move' and $d_o =$ 'do not move'
end procedure

procedure LINESEARCH$_t$($\mathbf{p}$, d, S, $\mathcal{M}$)
  $steps \leftarrow$ fixed set of step sizes
  for $step_j$ in $steps$ **do**
    $\mathbf{p}_j \leftarrow \text{updatePose}(\mathbf{p}, d, step_j)$
    $(R_{\text{F}}, R_{\text{VE}}, R_{\text{HE}}, R_{\text{BG}}) \leftarrow \text{render}(\mathbf{p}_j, \mathcal{M})$
    $score_j = \sum_{c \in \{\text{F,HE,VE,BG}\}} \sum_{i \in R_c} \log S_c^i$
  end for
  $j \leftarrow \arg\max_j score_j$
  return $\mathbf{p}_j$
end procedure

Illustrative examples showing the progress over time from the initial sensor pose to the finally obtained pose are given in Fig. 3.

Figure 3. Visualization of iteration steps taken by our algorithm for several scenes. Starting from the initial pose (first column) our method keeps iterating until it reaches the final pose (last column).

4. Experimental Results

To demonstrate the benefits of our approach, we first give an overview of the used benchmark and training data and then give results for artificial and real world scenarios.

4.1. Training and Evaluation Data

For training of these deep networks we used 50000 samples virtually generated from 95 images with known ground truth poses and computed the semantic segmentation for each image. To generate these samples, we added random noise on the ground truth poses, sampled from a uniform distribution: We sampled the location noise in the interval $[-10m; +10m]$ and the rotation noise in the interval $[-5°; +5°]$.

If the distance between the ground truth pose and the random pose was smaller than a threshold the desired output was set to the 'do not move' class; otherwise, we set it to the discretized direction closest to the direction between the ground truth and the random pose. The corresponding 2.5D models for each image were downloaded from OpenStreetMap, and we registered the images into the 3D world manually to obtain the ground truth poses.

For testing our approach, we used an extended version of the dataset proposed in [1] consisting of 40 images, where the orientation error of the sensors varies from $0.25°$ to $49°$; the location error varies from $0.25m$ to $23m$.

4.2. Converging from a Close Initial Estimate

Fig. 4 shows several examples of applying our algorithm for an initial pose that is within a radius of $5m$ from the ground truth location, and an orientation in the range of $[-5°; +5°]$.

4.3. Converging from an Estimate Provided by Real Sensors

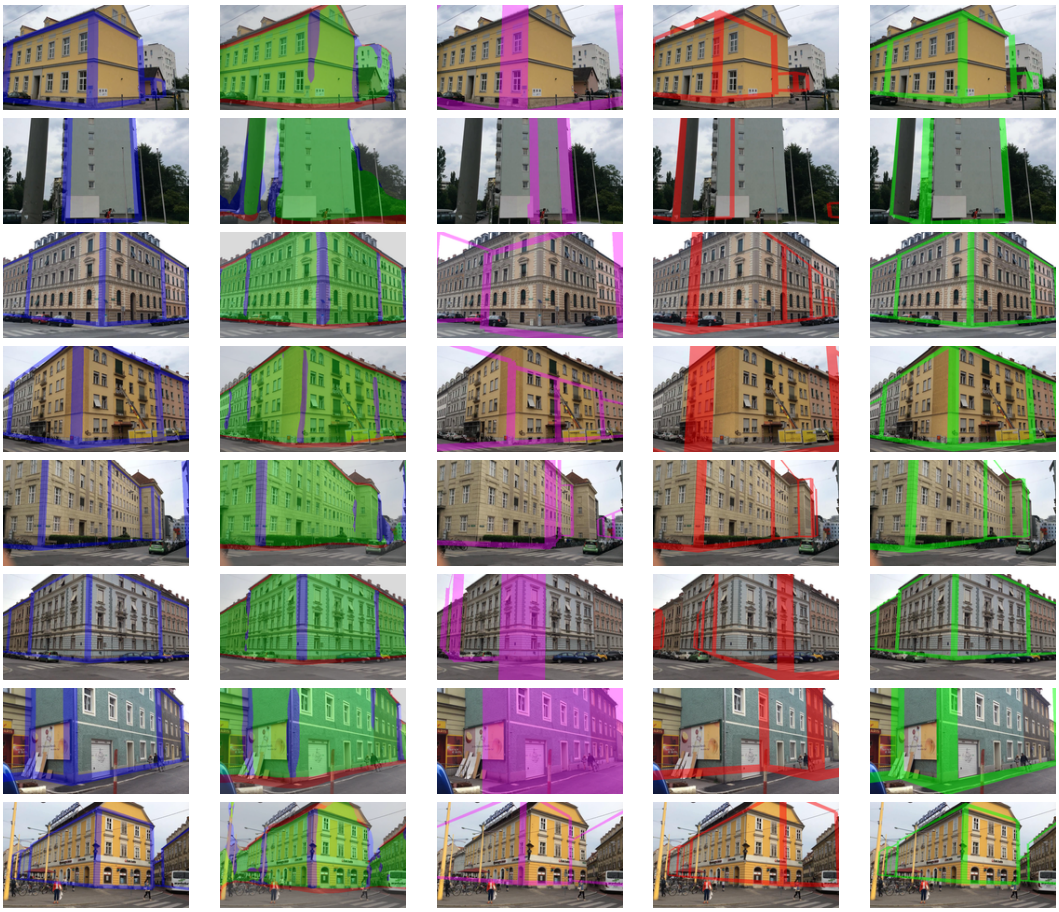
Real sensors can provide measurements with very large errors, in the order of 25 meters and 50 degrees. This makes convergence hard, since comparing the rendering from such a noisy pose and the input image cannot provide meaningful information for a pose update. In such a case, our strategy is to sample initial poses around the pose predicted by the sensors. We then run our iterative algorithm from each of these initial poses and keep the best final pose according to the log-likelihood. Fig. 5 shows examples of sensor poses and our finally estimated poses. Starting from the sensor data, our method decreases the average orientation error from $11.3°$ to $3.2°$. At the same time, the positioning error decreases from $13.4m$ to $3.1m$. Fig. 6 shows the orientation and the position errors for each image for both, real sensor poses and poses obtained by our method.

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

Figure 4. Converging from a close initial estimate. (a) Test image with the ground truth pose overlaid, (b) segmented image, (c) noisy pose rendering used to initialize our algorithm, (d) pose found with our method.

5. Conclusion

We showed that we can train networks to predict good directions to refine poses. We believe that such an approach is general: It is useful when it is not possible to differentiate an objective function as it is the case for our problem with the image likelihood, or when it is not clear which objective function should be optimized to reach a desired goal. Another advantage of this approach is that the training set can be augmented very easily, by generating estimates around real data: In our case, we could easily sample poses around the sensor poses, but this sampling strategy will also work for other problems.



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

Figure 5. Converging from an estimate provided by real sensors. (a) Test image with the ground truth pose overlayed, (b) segmented image, (c) real sensor pose, (d) pose where the optimization started the search to find the best estimated pose, (e) final pose found by our method.

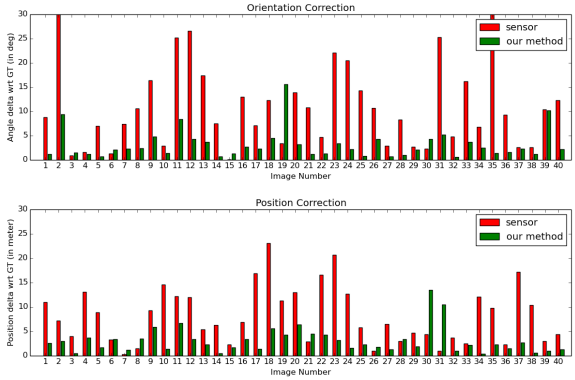


Figure 6. Orientation and position errors for sensor poses and poses obtained by our method.

Acknowledgment

This work was funded by the Christian Doppler Laboratory for Semantic 3D Computer Vision.

References

[1] C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg, and V. Lepetit. Instant Outdoor Localization and SLAM Initialization from 2.5D Maps. In *International Symposium on Mixed and Augmented Reality*, 2015.

[2] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys. Leveraging Topographic Maps for Image to Terrain Alignment. In *3DimPVT*, 2012.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[4] M. Blaha, C. Vogel, A. Richard, J. D. Wegner, T. Pock, and K. Schindler. Large-scale semantic 3d reconstruction: An adaptive multi-resolution model for multi-class volumetric labeling. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[5] T. Cham, A. Ciptadi, W. Tan, M. Pham, and L. Chia. Estimating Camera Pose from a Single Urban Ground-View Omnidirectional Image and a 2D Building Outline Map. In *Conference on Computer Vision and Pattern Recognition*, 2010.

[6] H. Chu, A. Gallagher, and T. Chen. GPS Refinement and Camera Orientation Estimation from a Single Image and a 2D Map. In *IEEE Workshop on Mobile Vision*, 2014.

[7] A. Cohen, A. G. Schwing, and M. Pollefeys. Efficient Structured Parsing of Facades Using Dynamic Programming. In *Conference on Computer Vision and Pattern Recognition*, 2014.

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[9] P. David and S. Ho. Orientation Descriptors for Localization in Urban Environments. In *International Conference on Intelligent Robots and Systems*, 2011.

[10] S. Feiner, B. MacIntyre, T. Höllerer, and A. Webster. A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. *Personal and Ubiquitous Computing*, 1(4):208–217, 1997.

[11] R. Gadde, R. Marlet, and N. Paragios. Learning Grammars for Architecture-Specific Facade Parsing. Research report, September 2014.

[12] V. Jampani, R. Gadde, and P. V. Gehler. Efficient Facade Segmentation Using Auto-Context. In *IEEE Winter Conference on Applications of Computer Vision*, 2015.

[13] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *International Conference on Computer Vision*, 2015.

[14] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2015.

[15] A. Martinović and L. Van Gool. Hierarchical Co-Segmentation of Building Facades. In *3DV*, 2014.

[16] N. Meierhold and A. Schmich. Referencing of images to laser scanner data using linear features extracted from digital images and range images. *International Society for Photogrammetry and Remote Sensing*, XXXVIII(3/W8):164–170, 2009.

[17] P. Meixner, A. Wendel, H. Bischof, and F. Leberl. Building Façade Separation in Vertical Aerial Images. *International Society for Photogrammetry and Remote Sensing*, I-3:239–243, 2012.

[18] S. Ramalingam, S. Bouaziz, and P. F. Sturm. Pose Estimation Using Both Points and Lines for Geo-Localization. In *International Conference on Robotics and Automation*, 2011.

[19] H. Riemenschneider, U. Krispel, W. Thaller, M. Donoser, S. Havemann, D. W. Fellner, and H. Bischof. Irregular Lattices for Complex Shape Grammar Facade Parsing. In *Conference on Computer Vision and Pattern Recognition*, 2012.

[20] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

[21] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[22] T. Sattler, B. Leibe, and L. Kobbelt. Improving Image-Based Localization by Active Correspondence Search. In *European Conference on Computer Vision*, 2012.

[23] G. Schindler, M. A. Brown, and R. Szeliski. City-Scale Location Recognition. In *Conference on Computer Vision and Pattern Recognition*, 2007.

[24] A. Taneja, L. Ballan, and M. Pollefeys. Registration of Spherical Panoramic Images with Cadastral 3D Models. In *3DimPVT*, 2012.

[25] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios. Segmentation of Building Facades Using Procedural Shape Priors. In *Conference on Computer Vision and Pattern Recognition*, 2010.

[26] T. Tieleman and G. Hinton. Lecture 6.5-Rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude. *Coursera: Neural Networks for Machine Learning*, 2012.

[27] G. Vaca-Castano, A. R. Zamir, and M. Shah. City Scale Geo-Spatial Trajectory Estimation of a Moving Camera. In *Conference on Computer Vision and Pattern Recognition*, 2012.

[28] V. Vineet, O. Miksik, M. Lidegaard, M. Nießner, S. Golodetz, V. A. Prisacariu, O. Kähler, D. W. Murray, S. Izadi, P. Perez, and P. H. S. Torr. Incremental Dense Semantic Stereo Fusion for Large-Scale Semantic Scene Reconstruction. In *International Conference on Robotics and Automation*, 2015.

[29] A. Wendel, M. Donoser, and H. Bischof. Unsupervised Facade Segmentation Using Repetitive Patterns. In *DAGM Symposium on Pattern Recognition*, 2010.

[30] K. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned Invariant Feature Transform. In *European Conference on Computer Vision*, 2016.

[31] A. R. Zamir and M. Shah. Accurate Image Localization Based on Google Maps Street View. In *European Conference on Computer Vision*, 2010.