Never Get Lost Again: Vision Based Navigation Using StreetView Images

Aparna Taneja(✉), Luca Ballan, and Marc Pollefeys

ETH Zurich, Zurich, Switzerland
aparna.taneja@gmail.com

Abstract. In this paper we propose a simple and lightweight solution to estimate the geospatial trajectory of a moving vehicle from images captured by a cellphone exploiting the map and the imagery provided by Google Streetview.

Images are intelligently compared against the streetview data, and a recursive Bayesian framework is applied to perform continuous localization of the vehicle inside the discrete structure of the streetview graph.

Experiments run on a dataset 10.7 km long, show that the system is able to infer its position and orientation despite the low resolution and limited field of view offered by an off the shelf consumer device. Our method shows to be robust with respect to significant changes in appearance and structure of the environment across the images, obtaining an average accuracy of 13 m in position, and 16° in orientation.

1 Introduction

Consumer navigation devices, such as Tomtom and Garmin, are common tools that assist drivers during their journey, as they provide directions, and help in the navigation of complicated networks of streets. These devices are nowadays available in most of the vehicles, but they heavily rely on the Global Positioning System (GPS), which is in general not very precise (with an accuracy that is at best around 10 m), and its signal may be absent or perturbed, especially in the presence of skyscrapers or inside a tunnel. Although this is not much of an issue for humans, its impact can be catastrophic when used in the context of autonomous or semi-autonomous driving vehicles.

To cope for this, the robotics community has explored the usage of other kinds of sensors, like compasses and inertial sensors, which, not only help in localizing the vehicle, but also provide its orientation. This additional information is very important, and would enable novel types of visualizations, in the context of navigation. Imagine the possibility of seeing an image of the street aligned with the point of view of the driver before a turn at a complicated intersection, or near an exit on a highway: directions can be superimposed on this image to avoid making the wrong turn (see Fig. 1). Such augmented visualizations are only possible if both the location and the orientation are estimated correctly. This is not possible using conventional sensors, since their measurements are highly affected by the environment. For instance, artificial magnetic fields, like

D. Cremers et al. (Eds.): ACCV 2014, Part V, LNCS 9007, pp. 99–114, 2015.
DOI: 10.1007/978-3-319-16814-2_7

Fig. 1. A device capturing images at low frame rate is mounted on a car driving around an urban environment. Our approach exploits the captured images and the streetview graph to track the movement of the vehicle on the map.

those created by power lines or tram lines, lead to noisy compass measurements with errors as high as 150° at times.

On the other hand, the vision community has developed several algorithms to infer the location of an image exploiting either large collections of geo-tagged images [1–3], or 3D models of urban environments [4–9]. While the availabilty of these 3D models is still restricted to a few cities in the world, large collection of images are becoming increasingly available thanks to services like Flickr and Panoramio. However, even for these collections, majority of the images are actually covering only popular/touristic locations around the world, and the number of images covering residential streets or highways for instance, is still very low.

Recently [10] proposed a method exploiting OpenStreetMaps (OSM) to perform continuous localization on a vehicle driving around a city. The key idea is to use the geometry of this map to align and localize the trajectories obtained from visual odometry [11–13].

Inspired by this work, we propose a method to use a similar map of the environment, in particular Google Streetview. Similarly to OSM, Google Streetview offers a graphical representation of the streets of the world as a network of nodes, each of which represents a specific location, and edges between these nodes represent their relative connectivity via streets. The level of detail of this map is however coarser compared to OSM. In fact, the streets are represented as piecewise linear segments sampled every 10 to 20 m, making it difficult to apply a curve matching based technique. To compensate for this lack of detail, we go one step further, and leverage also the image information available with this graph, i.e., the Google Streetview images.

Unlike the image collections provided by Flickr and Panoramio, Google Streetview offers a much broader and uniform coverage of the streets of the world, although at a relatively sparse sampling rate. This sparsity prohibits the usage of most of the 3D reconstruction techniques used by the large-scale localization approaches based on 3D models.

In this paper we propose a simple and lightweight solution to estimate the geospatial trajectory of a moving vehicle from images captured at 1 fps by an off the shelf consumer mobile device, such as a cellphone, mounted on the windshield

of the vehicle. We formulate the problem as a recursive Bayesian estimation of the position and the orientation of the vehicle using as observation the monocular images captured by the device, and the related compass measurements.

In contrast to classic image retrieval based techniques, we exploit the fact that our query images can be aligned almost perfectly with images in the database using a similar concept as proposed in Video Compass [14]. This allows us to maintain low computational requirements, and hence our solution can be easily ported as a client-server application on a mobile device. Therefore, for a person driving from one point to another, even if the GPS loses reception at some point, our system will help the driver localize and orient himself in the environment, by comparing images taken from the phone with those on the database.

In contrast to [10], which showed impressive results on stereo odometry we do not perform structure from motion and use only monocular videos, which was shown to be a challenging scenario even with their approach. Moreover, since we use the additional information provided by streetview images, our approach would in principle also work in a Manhattan world with a monotonous grid like structure.

2 Related Work

Most of the vision based localization approaches formulate the problem as an image retrieval task, typically using bag-of-words representations [15] or vocabulary trees [16] to efficiently search through large datasets of geo-tagged images. These methods rely on pure occurrence-based statistics to retrieve the geolocation of a query image [3,17–20]. In particular, [17] localizes videos taken from the web using Google streetview imagery. However, such methods in general fail in cases where the relative locations of the features on the image are also important. To cope for this, methods like [20–24] perform geometric consistency between the query image and the top ranked matching images from the database. This however becomes quite inefficient in case of repetitive urban structures or high fractions of mismatches, increasing the computationally complexity. Moreover, since these methods rely on RANSAC, their parameters need to be tuned precisely for each scene, as also observed in [25]. In our approach, we also take into account the relative location of the features on the images, but at the same time we aim at a simple lightweight solution, such that the computational load on both the client and the server side is minimal, and not much information has to be stored on the client side.

Methods like [6,8,26,27] instead, first recover the 3D structure of the images in the database and then perform a 2D-to-3D matching with the query image. If the number of inliers is higher than a certain threshold, the image is considered to be localized. 3D reconstruction however, is in itself an extremely difficult problem to solve, and it is even more challenging in a sparse dataset like Google Streetview.

In contrast, [10] performs continuous localization by aligning the trajectory of the vehicle obtained from visual odometry with a map of the environment.

Not only does this method require a highly detailed and accurate map of the environment, but it is also constrained by the fact that the images need to be captured with a sufficient density such that structure from motion can be applied, i.e., enough features can be matched across images.

Our approach instead, performs continuous localization on a coarser graphical representation of the environment using images captured on an average every 7 m by a phone camera with a limited field of view. Hence we cannot rely on visual odometry or curve matching techniques.

3 Algorithm

We model the map of the environment as a streetview graph (V, E), where each node $v \in V$ represents a specific location in the environment (stored as latitude and longitude coordinates), and each edge $e \in E$ between two nodes, indicates that the corresponding locations are connected by a street (see Fig. 1(right)). At each location v, a spherical panoramic image is available, representing the scene at that location. Let P_v denote this image. All of these panoramas are assumed to be aligned with respect to the north direction, and to have fixed roll and pitch angles relative to the tangent plane of the street.

We model the state of the car at time t using two random variables, $s_t \in V$ and $\rho_t \in S^1$, indicating respectively the position and the orientation of the car on the map. While the position is represented discretely as a node index on the streetview graph (V, E), the direction of motion ρ_t is represented as a unit vector in the x-y coordinates of the map. ρ_t is therefore a continuous quantity not necessarily indicating a valid traveling direction on the streetview graph as one might initially assume, i.e., ρ_t does not in general belong to E. This choice is made to account for changes of lane, U-turns, intersections, or in general any motion which is not modeled by the streetview graph.

Our algorithm tracks the state of the car at each time instance on the basis of the images captured by the device and their related compass measurements. Tracking is initialized with s_0 being the starting point of the car journey or being the last position measured by the GPS when the signal was available. The orientation ρ_0 is initialized as being equiprobable over all S^1. In all the subsequent time instances, our algorithm computes a probability map over all the possible car positions and orientations on the map. Precisely, it computes $P(s_t, \rho_t \mid I_t, c_t)$, i.e., the probability of each pair (s_t, ρ_t) given, as observations, the image I_t captured by the mobile device, and the corresponding compass measurement c_t, both measured at time t. Every time a new image is acquired, this probability map is updated on the basis of the new observations and a motion model. The best estimate for the car position is then obtained by selecting the state with maximum probability (maximum a posteriori).

While our approach broadly resembles the particle filtering algorithm, it is not the same, since no approximation on the posterior probability is made and no re-sampling is used. In fact, we exploit the already discrete nature of our model (see Sect. 3.3) to perform an exhaustive inference over all the possible states of

the car. At each time instance, all pairs (s_t, ρ_t) with probability different than 0, are stored as an array, and evaluated at the next time instance.

This makes our approach more robust and capable of recovering from tracking failures since it stores all the possible states of the car, even the least probable ones, helping in scenarios where, after some observations, these states turn out to be the correct ones.

3.1 Motion Model

The motion model provides us with a speculation on the position of the car at time t given the position and orientation probabilities $P(s_{t-1}, \rho_{t-1})$ at time $t-1$. It also provides time continuity on our inference, allowing us to cope with situations where the observations (I_t, c_t) are missing or not informative. This is the case when there are strong occlusions in the image, or when there are similar buildings in a row creating ambiguity on the correct location along the street.

We chose to use a constant speed motion model on the streetview graph. The constant speed motion model is defined in literature for continuous spaces, like $\mathbb{R}^2$ or $\mathbb{R}^3$, therefore some adjustments need to be made to make it work on the discrete space of a graph. Precisely, we first assume s_t to be a Markov chain of order one. Therefore,

$$P(s_t) = \sum P(s_t \mid s_{t-1}, \rho_{t-1}) P(s_{t-1}, \rho_{t-1}) \tag{1}$$

where the sum is intended to be over all $V \times S^1$, i.e., over all the possible positions and orientations (s_{t-1}, ρ_{t-1}), for which the probability $P(s_{t-1}, \rho_{t-1})$ is greater than 0. The probability map $P(s_{t-1}, \rho_{t-1})$ is the one provided by the algorithm at time step $t-1$, while $P(s_t \mid s_{t-1}, \rho_{t-1})$ defines the motion model and is described in the following paragraphs.

Precisely, if the car at time $t-1$ is observed to be at position s_{t-1} with an orientation ρ_{t-1}, it is likely that it is moving on the edge $e \in E$ of the streetview graph that is most parallel to the direction ρ_{t-1}. Therefore, at time t, the car must be on one of the nodes reachable from s_{t-1} through the edge e (see Fig. 2(left)). Note that this is independent from the orientation of the other edges along the path connecting s_t and s_{t-1}, since the car orientation ρ might have changed significantly from time $t-1$ to time t. We define the probability of reaching a specific node s_t on this path as

$$P(s_t \mid s_{t-1}, e) = \mathcal{N}_T(d_e(s_t, s_{t-1}), \sigma_m) \tag{2}$$

where $d_e(s_t, s_{t-1})$ is the length of shortest path connecting the nodes s_t and s_{t-1} that passes through the edge e. In the formula, $\mathcal{N}_T(\cdot, \sigma_m)$ denotes the truncated Gaussian distribution centered at zero and truncated for values less than 0. In our implementation, we set the standard deviation σ_m to 12 m, corresponding to the assumption that, in 68 % of the cases, the car is within 12 m of s_{t-1}.

All these probabilities are combined as follows

$$P(s_t \mid s_{t-1}, \rho_{t-1}) = \sum\nolimits_{e \in inc(s_{t-1})} P(s_t \mid s_{t-1}, e) P(e \mid \rho_{t-1}) \tag{3}$$

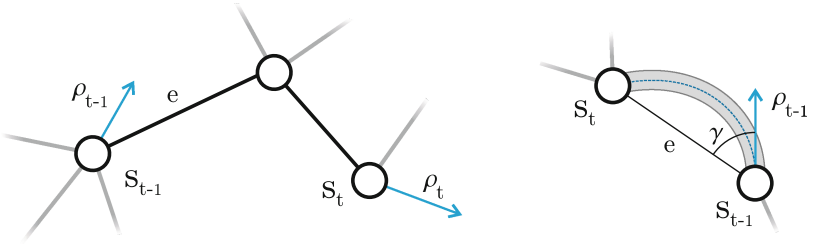

Fig. 2. Motion model on the streetview graph: (left) at position s_{t-1}, the car is likely to move on the edge e since it is the one most parallel to the heading direction ρ_{t-1}; (right) the actual path taken by the vehicle along the physical street might differ significantly from the path on the streetview graph (edge e).

where $inc\,(s_{t-1})$ represents the set of all edges $e \in E$ incident to node s_{t-1}. Using the Bayes' rule, we define $P\,(e \mid \rho_{t-1})$ as

$$P\,(e \mid \rho_{t-1}) = \frac{P\,(\rho_{t-1} \mid e)\,P\,(e)}{P\,(\rho_{t-1})} = \frac{1}{N} e^{k\cos(\gamma)} \tag{4}$$

where N is a normalization term ensuring that $\sum P\,(e \mid \rho_{t-1}) = 1$, γ is the angle between the edge e and the direction ρ_{t-1}, and where the concentration parameter k is set to 2.8 corresponding to a circular standard deviation of 40°. Precisely, here we implicitly assume that $P\,(e)$ is uniform, and $P\,(\rho_{t-1} \mid e)$ is distributed accordingly to a von-Mises distribution centered at the direction corresponding to the edge e. This is equivalent to assuming that the angle between the actual path taken by the vehicle, and the straight line connecting the two end points of that path, can vary along that path with a standard deviation of 40° (see Fig. 2(right)).

3.2 Observations

Given an image I_t, captured by the device at time t, we aim at inferring how likely is it, that the image was captured in the proximity of a streetview node v. This is performed by comparing the image I_t with the streetview panorama P_v corresponding to the node v.

In contrast to other image based localization techniques, we exploit the fact that, in our scenario, the image I_t and the panorama P_v are already well aligned, or at most they are aligned up to one degree of freedom. This is due to the fact that, in a setup where the device is assumed to be firmly attached to the windshield of the car or to its dashboard, the angle between the camera of the device and the driving direction is fixed over time. Since the driving direction is always parallel to the street, both the tilt and the roll angles of the camera are fixed with respect to the plane tangent to the street, and hence they need to be estimated only once, at the beginning of the journey. This is performed by capturing a few images from the device and by computing the pitch and the roll

Fig. 3. Street-level vanishing points at an intersection on a panoramic image P_v, and on an image I_t captured by the device. A perspective image P_v^α is extracted for each of the admissible angles, and compared with I_t.

angles which force the vertical vanishing point in I_t to lie on the image y-axis at infinity [28].

The yaw angle of the device instead, measured with respect to the north direction, changes over time and hence has to be estimated every time an image is captured. One might assume that, an initial guess for this orientation can be obtained from the compass. However, this sensor is very sensitive to any artificial magnetic fields present in the environment causing errors as high as 150°. Therefore an exhaustive search for the correct yaw angle needs to be performed.

Fortunately, in a practical scenario, this search can be limited to only those angles which make the forward street-level vanishing point on image I_t match one of the street-level vanishing points on the panoramic image P_v [14]. As an example, the image I_t in Fig. 3, can only have been captured at a yaw angle that makes its forward vanishing point match one of the three possible vanishing points in P_v, each of which correspond to a driving direction. We therefore extract a perspective image from P_v at each of these admissible yaw angles, and compare it to I_t. To be robust to changes in illumination, weather conditions and different camera settings across the images, this comparison is performed on a feature space. Precisely, let P_v^α be the perspective image extracted from panorama P_v at yaw angle α using the same intrinsic parameters as those of image I_t (these are assumed to be known a priori). We subdivide both I_t and P_v^α into blocks of size 30 by 30 pixels, and compute color and gradient descriptors for each of these blocks. We then compare corresponding blocks in each image, and sum up the results of these comparisons to obtain a score indicating how similar I_t and P_v^α are. Precisely, let $d_z\left(i, I_t\right)$ denote the descriptor of type z computed for the block i in image I_t, and let $d_z\left(i, P_v^\alpha\right)$ denote the corresponding descriptor computed on P_v^α. The similarity measure $o_t^{v,\alpha}$ between image I_t and image P_v^α, is then defined as

$$o_t^{v,\alpha} = \sum_{i,z} w_i^z \left\| d_z\left(i, I_t\right) - d_z\left(i, P_v^\alpha\right) \right\|_2 \tag{5}$$

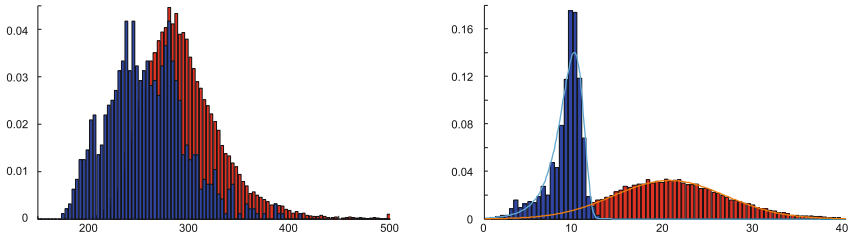

Fig. 4. Distribution of the scores $o_t^{v,\alpha}$ before (left), and after training (right) (Color figure online).

where w_i^z are constants weighing the contribution of each block and the relative influence between the color and the different gradient descriptors.

In our implementation, the color descriptor is simply computed as the average color among all the pixels in block i, hence it is a single triplet in the HSL space. Concerning the gradient descriptor instead, we used HOG [29] descriptors, and computed them as defined in the UoCTTI model [30] which includes the histogram of directed gradients, undirected gradients and a texture information. In addition, for each block we evaluate the entropy of the histogram of directed gradients, indicating how uniform is this distribution. We observed that this provides a sort of contextual information indicating whether the block depicts sky, buildings or road.

To compensate for small misalignments between I_t and P_v^α caused by the fact that, even though the images are aligned with the same orientation, they may have been captured on slightly different positions on the street (lat-long on the map), a window based comparison of the blocks is performed by comparing each block to the 8-neighbouring blocks and returning the minimum score.

Weights: Ideally the score $o_t^{v,\alpha}$ should be close to zero in case of similar images, however, in a first analysis (see Fig. 4(left)), assuming all weights w_i^z equal to 1, the distributions of the scores $o_t^{v,\alpha}$, in case of similar images (blue), and in case of not similar images (red), show a considerable overlap. This is due to the fact that, some feature types and some blocks contribute incoherently to the score, degrading its discriminative property. This happens, for instance, in areas of the image which are often occluded by cars and pedestrians, and in the areas that often contain objects close to the camera, and hence prone to registration artifacts. To cope with this, we learn the weights w_i^z minimizing the overlap between the two distributions. Precisely, we trained a linear Support Vector Machine [31] on a dataset of about 30k images, using 10-fold cross-validation. We then set our weights according to the resulting separating hyperplane.

Figure 4(right) shows the score distribution after the training. It is noticeable that, the score $o_t^{v,\alpha}$ is now sufficiently discriminative to tell us whether the image I_t is similar to P_v^α. To integrate this information into our probabilistic framework, we fit two standard distributions on these two histograms, precisely,

a Gaussian distribution for the non matching images, and a generalized extreme value distribution for the matching ones.

3.3 Posterior Probability and Tracking

Given the observed matching scores o_t and the compass measurement c_t, the posterior probability of the car state at time t can be written as

$$P(s_t, \rho_t \mid o_t, c_t) = \frac{1}{N} P(o_t, c_t \mid s_t, \rho_t) P(s_t, \rho_t) \tag{6}$$

$$= \frac{1}{N} P(o_t, c_t \mid s_t, \rho_t) P(\rho_t \mid s_t) P(s_t) \tag{7}$$

where the normalization term $N = P(o_t, c_t)$ is computed by ensuring that the sum $\sum P(s_t, \rho_t \mid o_t, c_t)$ over all pairs of nodes s_t and available directions ρ_t is equal to one. $P(s_t)$ is provided by the motion model, as described in Eq. 1. The probability $P(\rho_t \mid s_t)$ instead is assumed to be uniform over all the yaw angles admissible at node s_t, as described in Sect. 3.2. Precisely, let $\alpha_1, \ldots, \alpha_n$ be the set of all these admissible yaw angles, the probability density of ρ_t given s_t is therefore defined as,

$$P(\rho_t \mid s_t) = \frac{1}{n} \sum_{j=1}^{n} \delta(\rho_t - (\alpha_j - \Delta)) \tag{8}$$

where δ denotes the Dirac delta function, and Δ denotes the angle between the car heading direction and the device yaw direction. Note that Δ is fixed over time, and therefore it needs to be estimated only once, at the beginning of the journey. Equation 8 tells us that, $P(\rho_t \mid s_t)$ is different from zero if and only if ρ_t coincides with one of the admissible α in s_t, minus the correction Δ.

Concerning the likelihood $P(o_t, c_t \mid s_t, \rho_t)$, we assume independence between the compass measurements and the matching scores, therefore

$$P(o_t, c_t \mid s_t, \rho_t) = P(c_t \mid s_t, \rho_t) \prod_{v, \alpha_j} P\left(o_t^{v,\alpha_j} \mid s_t, \rho_t\right). \tag{9}$$

The compass measurements $P(c_t \mid s_t, \rho_t)$ are assumed to be affected by a circular Gaussian noise on S^1, that we approximate using a von-Mises distribution centered at the direction of motion ρ_t plus the correction Δ. Due to the high level of noise affecting this measurement, the standard deviation for this distribution was set to $\sigma_c = 60°$ in our experiments.

Concerning the generative model for the matching scores $P\left(o_t^{v,\alpha_j} \mid s_t, \rho_t\right)$ instead, we define it as

$$P\left(o_t^{v,\alpha_j} \mid s_t, \rho_t\right) = \begin{cases} \mathrm{Gev}(o_t^{v,\alpha_j}, \mu_+, \sigma_+, \xi_+) & s_t = v \wedge \rho_t = \alpha_j - \Delta \\ \mathcal{N}(o_t^{v,\alpha_j}, \mu_-, \sigma_-) & else \end{cases} \tag{10}$$

where $\mathrm{Gev}(\cdot, \mu_+, \sigma_+, \xi_+)$ and $\mathcal{N}(\cdot, \mu_-, \sigma_-)$ indicate the generalized extreme value distribution and the Gaussian distribution estimated in Sect. 3.2, for the matching images and the non matching ones, respectively.

Client-Server Framework: All previous operations (feature extraction, color conversion, color averaging and window based comparison) can be quickly performed on the GPU using shaders. Street-level vanishing points for each panorama in the database can be precomputed, and stored on the server side [32]. On the client side instead, only the streetview graph (V, E) is needed, not the panoramas P_v.

Every time a new image I_t is captured by the device, the forward vanishing point at the street level is estimated. We approximate this, by determining amongst all the vanishing points in the image, the one which is closest to the center of the image. Alternatively, this point can be easily tracked along the journey, as it always lies in the same region on the image, except when the car is turning. The descriptors for image I_t are then computed, and this information, along with the coordinates of the forward vanishing point, and the list of probable locations where the motion model is expecting the car to be, are sent to the server for evaluation. The required bandwidth for this communication is around 70 KB.

The server computes the scores o_t^{v,α_j} for each of the requested panoramas and each of the corresponding admissible yaw angles. During our experiments, the number of requested locations per image I_t was on an average 14. The server then sends back the results to the client which performs the inference updating the list of possible states as described above.

Time: For each phone image, vanishing point extraction takes 46ms while computing the descriptor takes 72 ms. Comparing this descriptor with an average of 14 locations and updating the tracked states takes under 0.001 ms.

4 Results

The proposed algorithm was evaluated on three different sequences captured by driving around an urban environment, covering a total distance of 10.7 km. Precisely, we mounted a Samsung Galaxy S4 on the windshield of a car at an estimated angle $\Delta = 20°$ with respect to the driving direction, as shown in Fig. 1. The phone captured images at 1 frame/sec, at a resolution of 960 by 540 pixels, and with a horizontal field of view of 60.3°. We drove at different times of the day and with moderate traffic conditions, at an average speed of 25 km/h with occasional peaks up to 65 km/h. Each image I_t was therefore captured at an average distance of 7 m, with peaks that went up to 18 m. Such a scenario would be quite challenging for a monocular structure from motion based method. Since we are using a low end consumer camera, the captured images suffered from rolling shutter distortion and motion blur. For comparison purposes, GPS information was also recorded.

The streetview graph (V, E) and the corresponding panoramic images were obtained using the Google StreetView API [33]. The error on this data is on an average around 3.7 m in the position, and 1.9° degrees in the orientation [34]. For the explored locations, the streetview data had an average sampling density of 14 m, and it was a few years old, showing structural and appearance changes in

Table 1. (Top) Statistics on the errors obtained in each of the evaluated sequences. (Bottom) Comparison with alternative approaches evaluated on sequence #1: (Geom) geometric verification based approach, (Feat) feature matching based comparison.

		ε_{graph} [nodes]			$\varepsilon_{\lvert\cdot\rvert_2}$ [m]			θ_{gps} [deg]		θ_{graph} [deg]	
		mean	median	std	mean	median	std	mean	median	mean	median
Our	**Seq. #1**	0.53	0.0	1.20	10.77	7.10	15.43	14.75	3.87	15.75	3.68
	Seq. #2	0.60	0.0	0.90	9.48	6.14	10.55	12.12	3.87	16.97	4.04
	Seq. #3	1.12	0.0	2.77	17.70	5.67	36.10	20.75	6.87	23.62	8.40
Alt	**Geom**	3.57	2.0	3.59	46.38	40.52	35.51	27.70	26.19	36.35	26.77
	Feat	4.23	2.0	4.70	47.10	34.46	40.46	18.03	4.27	19.58	3.90

the environment like new buildings, new paints/signs on buildings. Also different lighting conditions and seasonal changes (like changes in vegetation) were clearly visible compared to the phone images.

To quantitatively evaluate the performance of our method, we compared the obtained results against the GPS and the streetview graph. Table 1 reports the statistics of this comparison. In particular, we computed the Euclidean distance between our prediction and the GPS measurement, denoted as $\varepsilon_{|\cdot|_2}$ in the table. Since our estimate is constrained to be on the streetview graph, this error is biased by the discretization of the graph. Therefore, we also compute the length of the shortest path on the graph connecting our estimate and the node closest to the GPS position, ε_{graph} in the table. While the Euclidean distance is measured in meters, the distance on the graph is measured in number of nodes. Please note that, although $\varepsilon_{|\cdot|_2}$ might look high, one needs to account also for the sampling rate of the streetview data (14 m on average).

The performance on the orientation was evaluated in a similar way, by comparing our prediction with respect to the bearing direction measured by the GPS when the car was moving, θ_{gps} in the table, and also with the direction of the corresponding edge in the streetview graph, θ_{graph}. Again, this error has to be considered in light of the coarse representation of the streets in the map.

In general the algorithm performs well in all the three sequences. However, the third sequence was quite challenging, because the car drove through some regions which were not covered by the streetview graph. Therefore our estimate, due to this lack of connectivity, hovered around these regions until there were valid observations again, then the correct track was recovered gradually. This happened for 10 % of the frames, increasing the localization error.

Figure 5 shows some of the locations tested during our experiments. For each location, the figure provides the image captured by the phone I_t, the streetview image P_v^α corresponding to the location and the orientation estimated by our algorithm, and the streetview map zoomed in at the corresponding location. The green spheres on the map denote the possible car positions, with their size being proportional to their probabilities. The yellow sphere is the maximum a posteriori estimate with the related orientation shown as a red arrow. The cyan sphere represents the groundtruth GPS position with the related orientation measured by the compass (in blue), and measured by the GPS (in green).

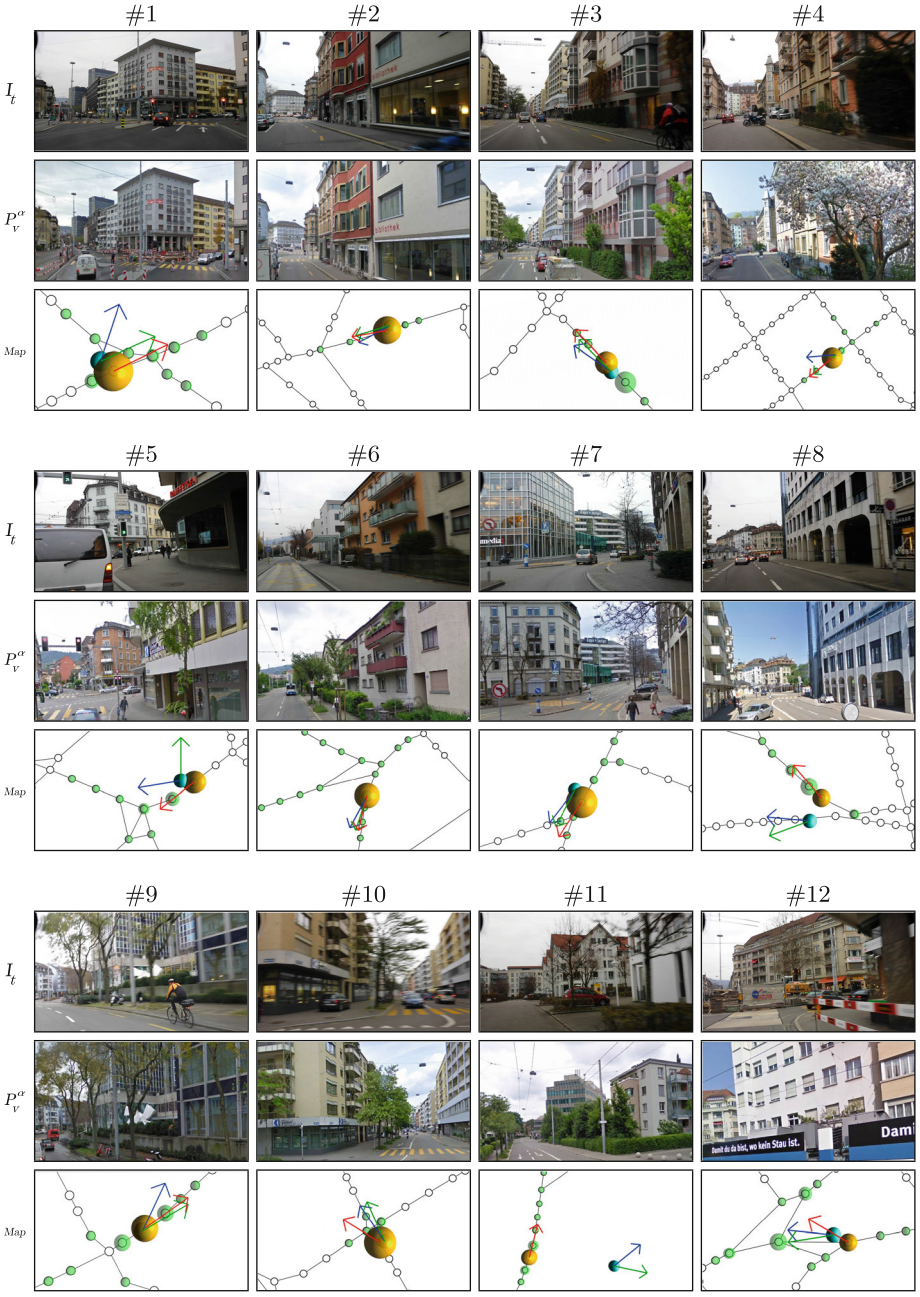

Fig. 5. Some of the evaluated locations: The map indicates the groundtruth GPS position (cyan sphere) displayed together with the bearing direction (green arrow), and the compass direction (blue arrow). Our prediction is displayed as a yellow sphere together with the estimated direction of motion (red arrow) (Color figure online).

Location #1 shows a typical inter ion, with streetside repair in progress in the panoramic image, while locations #2, #3, and #4 show examples of residential streets. In all these cases the alignment was quite accurate. In particular, the algorithm performs well in case #4, despite the strong change in vegetation causing a major occlusion. For both locations #3 and #4, the illumination between I_t and P_v was also quite different, since the latter was captured at a different time of the day. In cases #5, #6, and #7 instead, the scene had changed over time. Particulary, in location #5 the color of the front building and the signs on the building on the right had changed. Similarly in location #6, the building on the right had been repainted, and a new bus stop had been placed. Location #7 shows an example where an old building had been replaced completely with a new construction. Changes like this are quite challenging, despite this, the images were localized correctly. However, this may not happen when there are major changes on both sides of the street for instance. Location #8 shows an example where the GPS location was quite erroneous, whereas our algorithm was able to correctly localize the image on the graph. Such situations occur frequently, as the GPS is generally imprecise. Locations #9 and #10 show two scenarios where the images captured by the phone have significant motion blur. Despite this the algorithm was able to localize them accurately.

Failure Cases: All the above cases demonstrate the robustness of our method with respect to partial appearance and structural changes in the environment. However, failures occurred when the streetview data was extremely incoherent with respect to the current state. This is the case of locations #11 and #12 where in the first case, an entire street was missing, and in the second case, construction work in progress changed the layout of the intersection. Hence the algorithm lost track and recovered only after a while.

Computing the forward vanishing point on the phone image is only possible when enough structure is visible, i.e., when there are no strong occlusions and when the car is not facing only fronto-parallel buildings. However, in our experiments, this succeeded in 89 % of the cases. In the remaining cases, the motion model helps maintain the track.

Comparison with Prior Work: To compare our method with a geometric verification based technique, like the one proposed in [20], we extracted perspective images for each requested panorama at angles corresponding to the street directions in the graph. We then matched SURF [35] features between these images and the phone image I_t. Geometric verification was then performed by estimating the essential matrix relating each pair of images using RANSAC. Non-linear refinement was applied on the resulting matrix in order to minimize the reprojection error. The number of inliers was then recomputed on the basis of the new pose, and used as a feature to discriminate between similar and dissimilar images. We fit two Gaussian distributions on the basis of the statistics of these features. These were then used as generative model for the score $P\left(o_t^{v,\alpha_j} \mid s_t, \rho_t\right)$ in our framework. The fourth row in Table 1 reports the errors obtained with this method on sequence #1. The errors in the position are much higher, and since most of the matched features correspond to objects localized in a small region of the image, it increases the error in orientation as well.

To evaluate our approach with respect to a image retrieval based method, such as [2], we extracted perspective images from each of the requested panoramas at each of the admissible yaw angles, as in our approach. We then performed SURF matching by keeping only those correspondences satisfying the distance-ratio rule of [36]. This number was then used as a feature for our inference, as described before. Such a technique is in principle similar to a standard feature voting based retrieval technique, but deployed in conjunction with our motion model. The fifth row in Table 1, shows the results obtained for sequence #1. As expected, the error on orientation is low, since we used, as input, the already aligned images. The error on position instead, is higher, since this method neglects the geometric disposition of the features in the image. One should also consider that, in this case, the computational time was 7 times higher than in our approach.

5 Conclusions

We presented a method to perform continuous localization of a vehicle from images captured by a cellphone exploiting the map and the imagery provided by Google Streetview.

We formulated the problem as a recursive Bayesian estimation of the position and the orientation of the vehicle on the streetview graph. Differently from classic image retrieval based techniques, we exploit the fact that our query images can be aligned almost perfectly with the images in the database, keeping the computational requirements low.

Unlike sophisticated acquisition systems, we addressed a practical situation and perform continuous localization with a consumer mobile device with a limited field of view, low resolution and low frame rate. Despite the coarse representation of the streetview graph and its possible incoherence with the current structure and appearance of the environment, our algorithm achieved a good accuracy.

In principle, our method can be used on any datasets with street side imagery, such as those of Navteq, Microsoft etc., but we chose to use Google Streetview due to its universal coverage and free availability.

References

1. Schindler, G., Brown, M., Szeliski, R.: City-scale location recognition. In: CVPR (2007)
2. Zamir, A.R., Shah, M.: Accurate image localization based on google maps street view. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 255–268. Springer, Heidelberg (2010)
3. Hays, J., Efros, A.A.: Im2gps: estimating geographic information from a single image. In: CVPR (2008)
4. Baatz, G., Koser, K., Chen, D., Grzeszczuk, R., Pollefeys, M.: Leveraging 3D city models for rotation invariant place-of-interest recognition. IJCV **96**(3), 315–334 (2012)

5. Ramalingam, S., Bouaziz, S., Sturm, P., Brand, M.: Skyline2gps: localization in urban canyons using omni-skylines. In: IROS (2010)
6. Sattler, T., Leibe, B., Kobbelt, L.: Fast image-based localization using direct 2D-to-3D matching. In: ICCV (2011)
7. Irschara, A., Zach, C., Frahm, J.M., Bischof, H.: From structure-from-motion point clouds to fast location recognition. In: CVPR (2009)
8. Li, Y., Snavely, N., Huttenlocher, D., Fua, P.: Worldwide pose estimation using 3D point clouds. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part I. LNCS, vol. 7572, pp. 15–29. Springer, Heidelberg (2012)
9. Schindler, G., Krishnamurthy, P., Lublinerman, R., Liu, Y., Dellaert, F.: Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In: CVPR (2008)
10. Brubaker, M.A., Geiger, A., Urtasun, R.: Lost! leveraging the crowd for probabilistic visual self-localization. In: CVPR (2013)
11. Wu, C.: Towards linear-time incremental structure from motion. In: 3DV (2013)
12. Kaess, M., Ni, K., Dellaert, F.: Flow separation for fast and robust stereo odometry. In: ICRA (2009)
13. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: dense 3D reconstruction in real-time. In: IEEE Intelligent Vehicles Symposium (2011)
14. Kosecká, J., Zhang, W.: Video compass. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part IV. LNCS, vol. 2353, pp. 476–490. Springer, Heidelberg (2002)
15. Sivic, J., Zisserman, A.: Video google: a text retrieval approach to object matching in videos. In: ICCV (2003)
16. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: CVPR (2006)
17. Vaca, G., Zamir, A.R., Shah, M.: City scale geo-spatial trajectory estimation of a moving camera. In: CVPR (2012)
18. Torii, A., Sivic, J., Pajdla, T., Okutomi, M.: Visual place recognition with repetitive structures. In: CVPR (2013)
19. Cummins, M., Newman, P.: Fab-map: probabilistic localization and mapping in the space of appearance. Int. J. Robot. Res. **27**(6), 647–665 (2008)
20. Zhang, W., Kosecka, J.: Image based localization in urban environments. In: 3DPVT, pp. 33–40 (2006)
21. Galvez-Lopez, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Trans. Rob. **28**(5), 1188–1197 (2012)
22. Baatz, G., Köser, K., Chen, D., Grzeszczuk, R., Pollefeys, M.: Handling urban location recognition as a 2D homothetic problem. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 266–279. Springer, Heidelberg (2010)
23. Sunderhauf, N., Protzel, P.: Brief-gist closing the loop by simple means. In: IROS (2011)
24. Raguram, R., Tighe, J., Frahm, J.M.: Improved geometric verification for large scale landmark image collections. In: BMVC (2012)
25. Lee, G.H., Pollefeys, M.: Unsupervised learning of threshold for geometric verification in visual-based loop-closure. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
26. Lim, H., Sinha, S.N., Cohen, M.F., Uyttendaele, M.: Real-time image-based 6-DOF localization in large-scale environments. In: CVPR (2012)

27. Li, X., Wu, C., Zach, C., Lazebnik, S., Frahm, J.-M.: Modeling and recognition of landmark image collections using iconic scene graphs. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 427–440. Springer, Heidelberg (2008)
28. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2004)
29. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
30. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms (2008). http://www.vlfeat.org/
31. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: a library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)
32. Bazin, J.C., Pollefeys, M.: 3-line RANSAC for orthogonal vanishing point detection. In: IROS (2012)
33. https://developers.google.com/maps/documentation/streetview/
34. Taneja, A., Ballan, L., Pollefeys, M.: Registration of spherical panoramic images with cadastral 3D models. In: 3DIMPVT (2012)
35. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
36. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**, 91–110 (2004)