Lost! Leveraging the Crowd for Probabilistic Visual Self-Localization

Marcus A. Brubaker
TTI Chicago
mbrubake@cs.toronto.edu

Andreas Geiger
KIT & MPI Tübingen
geiger@kit.edu

Raquel Urtasun
TTI Chicago
rurtasun@ttic.edu

Abstract

*In this paper we propose an affordable solution to self-localization, which utilizes visual odometry and road maps as the only inputs. To this end, we present a probabilistic model as well as an efficient approximate inference algorithm, which is able to utilize distributed computation to meet the real-time requirements of autonomous systems. Because of the probabilistic nature of the model we are able to cope with uncertainty due to noisy visual odometry and inherent ambiguities in the map (*e.g.*, in a Manhattan world). By exploiting freely available, community developed maps and visual odometry measurements, we are able to localize a vehicle up to 3m after only a few seconds of driving on maps which contain more than 2,150km of drivable roads.*

1. Introduction

Self-localization is key for building autonomous systems that are able to help humans in everyday tasks. Despite decades of research, it is still an exciting open problem. In this paper we are interested in building *affordable* and *robust* solutions to self-localization for the autonomous driving scenario. Currently, the leading technology in this setting is GPS. While being a fantastic aid for human driving, it has some important limitations in the context of autonomous systems. Notably, the GPS signal is not always available, and its localization can become imprecise (*e.g.*, in the presence of skyscrapers, tunnels or jammed signals). While this might still be viable for human driving, consequences can be catastrophic for self-driving cars.

To provide alternatives to GPS localization, place recognition approaches have been developed. They assume that image or depth features from anywhere around the globe can be stored in a database, and cast the localization problem as a retrieval task. Both 3D point clouds [5, 7, 10, 20] and visual features [2, 3, 11, 15, 16, 24] have been leveraged to solve this problem. In combination with GPS, impressive results have been demonstrated (*e.g.*, the Google self-driving car). However, it remains unclear if maintaining an up-to-date world representations will be feasible given the computation, memory and communication requirements. Furthermore, these solutions are far from affordable as every corner of the world needs to be visited and updated constantly. Finally, privacy and security issues need to be considered as the recording and storage of such data is illegal in some countries.

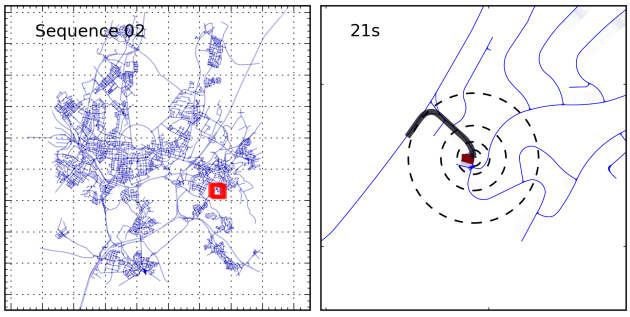

Figure 1. **Visual Self-Localization:** We demonstrate localizing a vehicle with an average accuracy of 3.1m within a map of $\sim 2,150$km of road using only visual odometry measurements and freely available maps. In this case, localization took less than 21 seconds. Grid lines are every 2km.

In contrast to the above mentioned approaches, here we tackle the problem of self-localization in places that we have never seen before. We take our inspiration from humans, which excel in this task while having access to only a rough cartographic description of the environment. We propose to *leverage the crowd*, and exploit the development of OpenStreetMap (OSM), a free community-driven map, for the task of vision-based localization. The OSM maps are detailed and freely available, making this an inexpensive solution. Moreover, they are more frequently updated than their commercial counterparts. Towards this goal, we derive a probabilistic map localization approach that uses visual odometry estimates and OSM data as the only inputs. We demonstrate the effectiveness of our approach on a variety of challenging scenarios making use of the recently released KITTI visual odometry benchmark [8]. As our experiments show, we are able to localize ourselves after only a few seconds of driving with an accuracy of 3 meters on a 18km^2 map containing $2,150$km of drivable roads.

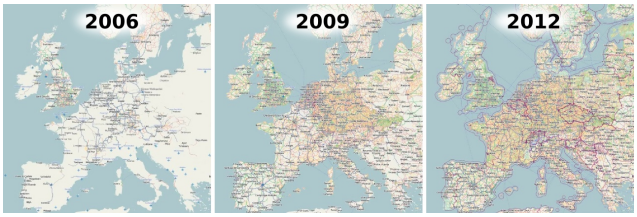

Figure 2. **Evolution of OpenStreetMap coverage from 2006-2012:** As of 2012, over 3 billion GPS track points have been added and 1.6 billion nodes / 150 million line segments have been created by the community. Here we use OSM maps and visual odometry estimates as the only inputs for localizing within the map.

2. Related Work

Early approaches for *map localization* [5, 7, 10, 20] make use of Monte Carlo methods and the Markov assumption to maintain a sample-based posterior representation of the agent's pose. However, they only operate locally without providing any global (geographic) positioning information and thus can not be applied to the problem we consider here. Furthermore, they are typically restricted to small-scale environments and low-noise laser-scan observations.

At a larger scale, *place recognition* methods localize [2, 11, 16, 24] or categorize [22, 23, 27] an image, given a database of geo-referenced images or video streams [3, 15]. While processing single landmarks is clearly feasible, creating an up-to-date "world database" seems impractical due to computational and memory requirements. In contrast, the maps used by our localization approach require only a few gigabytes for storing the whole planet earth[1].

Relative motion estimates can be obtained using *visual odometry* [19], which refers to generating motion estimates from visual input alone. While current implementations [1, 9, 14] demonstrate impressive performance [8], their incremental characteristics inevitably leads to large drift at long distances. Methods for *Simultaneous Localization And Mapping (SLAM)* [18, 25, 6] are able to reduce this drift by modelling the map using landmarks and jointly optimizing over poses and landmarks. Limitations in terms of speed and map size have been partially overcome, for example by efficient optimization strategies using incremental sparse matrix factorization [13] or the use of relative representations [17]. Furthermore, recent progress in *loop-closure* detection [4, 21, 26] has led to improved maps by constraining the problem at places which have been visited multiple times. However, SLAM methods can only localize themselves in maps that have been previously created with a similar sensor setup, hence strongly limiting their application at larger scales. In contrast, the proposed approach enables geographic localization and relies only on freely available map information (*i.e.*, OpenStreetMap). To our knowledge, ours is the first approach in this domain.

[1] http://wiki.openstreetmap.org/wiki/planet.osm

3. Visual Localization

We propose to use one or two roof-mounted cameras to self-localize a driving vehicle. The only other information we have is a map of the environment in which the vehicle is driving. This map contains streets as line segments as well as intersection points. We exploit visual odometry in order to obtain the trajectory of the vehicle. As this trajectory is too noisy for direct shape matching, here we propose a probabilistic approach to self-localization that employs visual odometry measurements in order to determine the instantaneous position and orientation of the vehicle in a given map. Towards this goal, we first define a graph-based representation of the map as well as a probabilistic model of how a vehicle can traverse the graph. For inference, we derive a filtering algorithm, which exploits the structure of the graph using Mixtures of Gaussians. In order to keep running times reasonable, we further propose techniques for limiting the complexity of the mixture models which includes an algorithm for simplifying the Gaussian Mixture models. We start our discussion by presenting the employed map information, followed by our probabilistic model.

3.1. The OpenStreetMap Project

Following the spirit of Wikipedia, Steve Coast launched the OpenStreetMap (OSM) project in 2004 with the goal of creating a free editable map of the world. So far, more than 800,000[2] users around the globe have contributed by supplying tracks from portable GPS devices, labeling objects using aerial imagery or providing local information. Fig. 2 illustrates the tremendous growth of OSM over the last years. Compared to commercial products like Google Maps, the provided data is more up-to-date, often includes more details (*e.g.*, street types, traffic lights, postboxes, trees, shops, power lines) and – most importantly – can be freely downloaded and used under the Open Database License. We extracted all crossings and drivable roads (represented as piece-wise linear segments) connecting them. For each street we additionally extract its type (*i.e.*, highway or rural) and the direction of traffic. By splitting each bi-directional street into two one-way streets and 'smoothing' intersections using circular arcs, we obtain a lane-based map representation, on which we define the vehicle state.

3.2. Lane-based Map Representation

We assume that the map data is represented by a directed graph where nodes represent street segments and edges define the connectivity of the roads. Roads which dead-end or run off the edge of the map are connected to a "sink" node. As mentioned above, we convert all street segments to one-way streets. An example of a map and corresponding graph representation is shown in Fig. 3 (left). Each street segment

[2] http://wiki.openstreetmap.org/wiki/stats

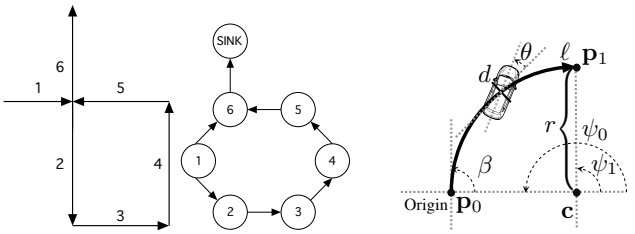

Figure 3. **Map Graph:** (left) A simple map and its corresponding graph representation. **Street Segment:** (right) Each street segment has a start and end position $\mathbf{p}_0$ and $\mathbf{p}_1$, a length ℓ, an initial heading of the street segment β and a curvature parameter $\alpha = \frac{\psi_1 - \psi_0}{\ell}$. For arc segments $\mathbf{c}$ is the circle center, r is the radius and ψ_0 and ψ_1 are the start and end angles of the arc. For linear segments, $\alpha = 0$.

is either a linear or a circular arc segment. The parameters of the street segment geometry are described in Fig. 3 (right). We define the position and orientation of a vehicle in the map in terms of the street segment u that the vehicle is on, the distance from the origin of that street segment d and the offset of the local street heading θ. The global heading of the vehicle is then $\theta + \beta + \alpha d$ and its position is $\frac{\ell-d}{\ell}\mathbf{p}_0 + \frac{d}{\ell}\mathbf{p}_1$ for a linear segment and $\mathbf{c} + r\mathbf{d}(\frac{\ell-d}{\ell}\psi_0 + \frac{d}{\ell}\psi_1)$ for a circular arc segment, with $\mathbf{d}(\theta) = (\cos\theta, \sin\theta)^T$.

3.3. State-Space Model

We define the state of the model at time t to be $\mathbf{x}_t = (u_t, \mathbf{s}_t)$ where $\mathbf{s}_t = (d_t, \hat{d}_{t-1}, \theta_t, \hat{\theta}_{t-1})^T$ and $\hat{d}_{t-1}$, $\hat{\theta}_{t-1}$ are the distance and angle at the previous time defined relative to the current street u_t. Visual odometry observations at time t, $\mathbf{y}_t$, measure the linear and angular displacement from time $t-1$ to time t. We thus model

$$p(\mathbf{y}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{y}_t|\mathbf{M}_{u_t}\mathbf{s}_t, \Sigma^{\mathbf{y}}_{u_t}) \tag{1}$$

where $\mathbf{M}_u = [\mathbf{m}_d, \mathbf{m}_\theta]^T$, $\mathbf{m}_d = (1, -1, 0, 0)^T$ and $\mathbf{m}_\theta = (\alpha_u, -\alpha_u, 1, -1)^T$. The curvature of the street, α_u, is necessary because the global heading of the vehicle depends on both d and θ. We factorize the state transition distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1}) = p(u_t|\mathbf{x}_{t-1})p(\mathbf{s}_t|u_t, \mathbf{x}_{t-1})$ in terms of the street transition probability $p(u_t|\mathbf{x}_{t-1})$, and the state transition model $p(\mathbf{s}_t|u_t, \mathbf{x}_{t-1})$. The state transition model is assumed to be Gauss-Linear, taking the form

$$p(\mathbf{s}_t|u_t, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{s}_t|\mathbf{A}_{u_t,u_{t-1}}\mathbf{s}_{t-1} + \mathbf{b}_{u_t,u_{t-1}}, \Sigma^{\mathbf{x}}_{u_t}) \tag{2}$$

with $\Sigma^{\mathbf{x}}_{u_t}$ the covariance matrix for a given u_t which is learned from data as discussed in Section 4. We use a second-order, constant velocity model for the change in d and a first order autoregressive model, *i.e.*, AR(1), for the angular offset θ. That is, $d_t = d_{t-1} + (d_{t-1} - \hat{d}_{t-2})$ plus noise, and $\theta_t = \gamma_{u_{t-1}}\theta_{t-1}$ plus noise where $\gamma_{u_{t-1}} \in [0, 1]$ is the parameter of the AR(1) model which controls the correlation between θ_t and θ_{t-1}. In practice, we found these models to be both simple and effective. Because the components of $\mathbf{s}_t$ are relative to the current street, u_t, when $u_t \neq u_{t-1}$ the state transition model must be adjusted so that $\mathbf{s}_t$ becomes relative to u_t. Both d_t and $\hat{d}_{t-1}$ must have $\ell_{u_{t-1}}$ subtracted, and $\hat{\theta}_{t-1}$ needs to be updated so that $\hat{\theta}_{t-1}$ relative to u_t has the same *global* heading as θ_{t-1} relative to u_{t-1}. The above model can then be expressed as

$$\mathbf{A}_{u_t,u_{t-1}} = \begin{bmatrix} 2 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \gamma_{u_t} & 0 \\ 0 & \alpha_{u_{t-1}} - \alpha_{u_t} & 1 & 0 \end{bmatrix} \tag{3}$$

$$\mathbf{b}_{u_t,u_{t-1}} = \begin{cases} -(\ell_{u_{t-1}}, \ell_{u_{t-1}}, 0, \theta_{u_t,u_{t-1}})^T & u_t \neq u_{t-1} \\ (0,0,0,0)^T & u_t = u_{t-1} \end{cases} \tag{4}$$

where $\theta_{u_t,u_{t-1}} = \beta_{u_t} - (\beta_{u_{t-1}} + \alpha_{u_t}\ell_{u_{t-1}})$ is the angle between the end of u_t and the beginning of u_{t-1}.

The street transition probability $p(u_t|\mathbf{x}_{t-1})$ defines the probability of transitioning onto the street u_t given the previous state $\mathbf{x}_{t-1}$. We use the Gaussian transition dynamics to define the probability of changing street segments, *i.e.*,

$$p(u_t|\mathbf{x}_{t-1}) = \xi_{u_t,u_{t-1}} \int_{\ell_{u_{t-1}}}^{\ell_{u_{t-1}}+\ell_u} \mathcal{N}(x|\mathbf{a}_d^T\mathbf{s}_{t-1}, \mathbf{a}_d^T\Sigma^{\mathbf{x}}_{u_{t-1}}\mathbf{a}_d)dx \tag{5}$$

where $\mathbf{a}_d = (2, -1, 0, 0)$,

$$\xi_{u_t,u_{t-1}} = \begin{cases} 1 & u_t = u_{t-1} \\ \frac{1}{|\mathbf{N}(u_j)|} & u_t \in \mathbf{N}(u_{t-1}) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

and $\mathbf{N}(u)$ is the set of streets to which u connects.

As short segments cannot be jumped over in a single time step, we introduce "leapfrog" edges which allow the vehicle to move from u_{t-1} to any u_t to which there exists a path in the graph. To handle this properly, we update the entries of $\mathbf{b}_{u_t,u_{t-1}}$ to consider transitioning over a longer path and $\xi_{u_t,u_{t-1}}$ is the product ξ along the path. As the speed of the vehicle is assumed to be limited, we need to add edges only up to a certain distance. Assuming a top speed of around 110km/h and observations every second, we add leapfrog edges for paths of up to 30m.

3.4. Inference

Given the above model we wish to compute the filtering distribution, $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. We can write the posterior using the product rule as $p(\mathbf{x}_t|\mathbf{y}_{1:t}) = p(\mathbf{s}_t|u_t, \mathbf{y}_{1:t})p(u_t|\mathbf{y}_{1:t})$, where $p(u_t|\mathbf{y}_{1:t})$ is a discrete distribution over streets and $p(\mathbf{s}_t|u_t, \mathbf{y}_{1:t})$ is a continuous distribution over the position and orientation on a given street. We choose to represent $p(\mathbf{s}_t|u_t, \mathbf{y}_{1:t})$ using a Mixture of Gaussians, *i.e.*,

$$p(\mathbf{s}_t|u_t, \mathbf{y}_{1:t}) = \sum_{i=1}^{N_{u_t}} \pi^{(i)}_{u_t}\mathcal{N}(\mathbf{s}_t|\mu^{(i)}_{u_t}, \Sigma^{(i)}_{u_t}) \tag{7}$$

Algorithm 1 Filter

Input: Posterior at $t-1$, $\{P_u^{t-1}, \mathcal{M}_u^{t-1}\}$, and observation, $\mathbf{y}_t$
Initialize mixtures, $\mathcal{M}_u^t \leftarrow \varnothing$, for all u
for all streets u_{t-1} **do**
  for all streets u_t reachable from u_{t-1} **do**
    for $k = 1, \ldots, |\mathcal{M}_{u_{t-1}}^{t-1}|$ **do**
      if $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$ is approx. constant **then**
        Analytically approx. $c_{pred}\mathcal{N}(\mu_{pred}, \Sigma_{pred})$
      else
        Sample to compute $c_{pred}\mathcal{N}(\mu_{pred}, \Sigma_{pred})$
      Incorporate $\mathbf{y}_t$ to compute $c_{upd}\mathcal{N}(\mu_{upd}, \Sigma_{upd})$
      Add $\mathcal{N}(\mu_{upd}, \Sigma_{upd})$ to $\mathcal{M}_{u_i}^t$ with weight c_{upd}
for all streets u **do**
  Set P_u^t to the sum of the weights of mixture $\mathcal{M}_u^t$
  Normalize the weights of mixture $\mathcal{M}_u^t$
Normalize P_u^t so that $\sum_u P_u^t = 1$.
Return: Posterior at t, $\{P_u^t, \mathcal{M}_u^t\}$

where N_{u_t} is the number of components for the mixture associated with u_t and $\mathcal{M}_{u_t}^t = \{\pi_{u_t}^{(i)}, \mu_{u_t}^{(i)}, \Sigma_{u_t}^{(i)}\}_{i=1}^{N_{u_t}}$ are the parameters of the mixture for u_t. This is a general and powerful representation but still allows for efficient and accurate inference. Assuming independent observations given the states and that the state transitions are first order Markov, we write the filtering distribution recursively as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \int \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \tag{8}$$

which, after factoring $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$, gives

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \sum_{u_{t-1}} \frac{P_{u_{t-1}}}{Z_t} p(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{s}_t|u_t, u_{t-1}, \mathbf{s}_{t-1}) \times p(u_t|u_{t-1}, \mathbf{s}_{t-1}) p(\mathbf{s}_{t-1}|u_{t-1}, \mathbf{y}_{1:t-1}) d\mathbf{s}_{t-1} \tag{9}$$

where $P_{u_{t-1}} = p(u_{t-1}|\mathbf{y}_{1:t-1})$ and $Z_t = p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$.

Substituting in the mixture model form of $p(\mathbf{s}_{t-1}|u_{t-1}, \mathbf{y}_{1:t-1})$, and the model transition dynamics the integrand in the above equation becomes

$$\sum_{i=1}^{N} \pi^{(i)} \int p(u_t|u_{t-1},\mathbf{s}_{t-1}) \mathcal{N}(\mathbf{s}_t|\mathbf{A}\mathbf{s}_{t-1} + \mathbf{b}, \Sigma^{\mathbf{x}}) \times \mathcal{N}(\mathbf{s}_{t-1}|\mu^{(i)}, \Sigma^{(i)}) d\mathbf{s}_{t-1} \,. \tag{10}$$

In general, the integral in Eq. (10) is not analytically tractable. However, if $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$ were constant the integral could be solved easily. In our model $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$ is the Gaussian CDF and has a sigmoidal shape. Because of this, it is approximately constant everywhere except near the transition point of the sigmoid. We determine whether $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$ can be considered constant and, if so, use an analytical approximation. Otherwise, we use a Monte Carlo approximation, drawing samples from $\mathcal{N}(\mathbf{s}_{t-1}|\mu^{(i)}, \Sigma^{(i)})$. Finally the observation $\mathbf{y}_t$ is incorporated by multiplying two Gaussian PDFs. This algorithm can also be parallelized by assigning subsets of streets to different threads, a fact which we exploit to achieve real-time performance. Appendix A gives more details and the filtering process is summarized in Algorithm 1.

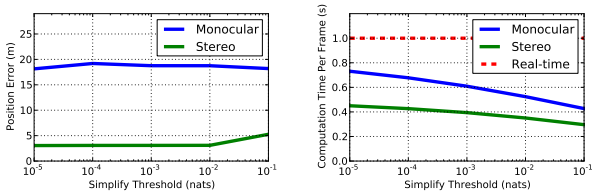

Figure 4. **Simplification Threshold:** Impact of the simplification threshold ϵ on localization accuracy (left) and computation time (right). We use $\epsilon = 10^{-2}$ for all other experiments.

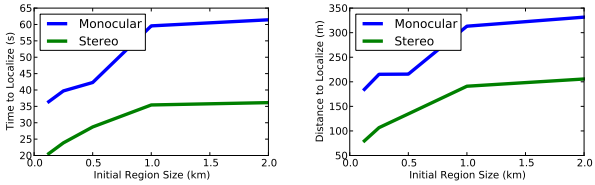

Figure 5. **Map Size:** Driving time (left) and distance travelled (right) before localization as a function of the map size.

3.5. Managing Posterior Complexity

The previous section provides a basic algorithm to compute the filtering distributions recursively. Unfortunately, it is impractical as the complexity of the posterior (*i.e.*, the number of mixture components) grows exponentially with time. To alleviate this, we propose three approximations which limit the resulting complexity of the posterior. We have found these approximations to work well in practice and to significantly reduce computational costs.

First, for each pair of connected streets, the modes that transition from u_{t-1} to u_t are all likely similar. As such, all of the transitioned modes are replaced with a single component using moment matching. Second, eventually most streets will have negligible probability. Thus, we truncate the distribution for streets whose probability $p(u_t|\mathbf{y}_{1:t})$ is below a threshold and discard their modes. We use a conservative threshold of 10^{-50}. Finally, the number of components in the posterior grows with t. Many of those components will have small weight and be redundant. To prevent this from happening, we run a mixture model simplification procedure when the number of modes on a street segment exceeds a threshold. This procedure removes components and updates others while keeping the KL divergence below a threshold ϵ. Details of this approximation can be found in Appendix B, and the effects of varying the maximum allowed KL divergence, ϵ, are investigated in the experiments.

		00	01	02	03	04	05	06	07	08	09	10	Average
Position Error	M	15.6m	*	8.1m	18.8m	*	5.6m	*	15.5m	45.2m	5.4m	*	18.4m
	S	2.1m	3.8m	4.1m	4.8m	*	2.6m	*	1.8m	2.4m	4.2m	3.9m	3.1m
	G	1.8m	2.5m	2.2m	6.9m	*	2.7m	*	1.5m	2.0m	3.8m	2.5m	2.4m
	O	0.8m	1.3m	1.0m	2.5m	3.9m	1.3m	1.0m	0.6m	1.1m	1.2m	1.1m	1.44m
Heading Error	M	2.0°	*	1.5°	2.4°	*	2.0°	*	1.3°	10.3°	1.6°	*	3.6°
	S	1.2°	2.7°	1.3°	1.6°	*	1.4°	*	1.9°	1.2°	1.3°	1.3°	1.3°
	G	1.0°	1.0°	0.8°	1.4°	*	1.2°	*	1.5°	1.0°	0.9°	1.0°	1.0°

Table 1. **Sequence Errors:** Average position and heading errors for 11 training sequences. "M" and "S" indicate monocular and stereo odometry, "G" GPS-based odometry and "O" is the oracle error, *i.e.*, the error from projecting the GPS positions onto the map. Chance performance is 397m. All averages are computed over localized frames (see text) and "*" indicates sequences which did not localize.

4. Experimental Evaluation

To evaluate our approach in realistic situations, we performed experiments on the recently released KITTI benchmark for visual odometry [8]. We utilize the 11 training sequences for quantitative evaluation (where ground truth GPS data is available), and perform qualitative evaluation on both training and test sequences (see Supplemental Material). This results in 39.2km of driving in total. The visual odometry input to our system is computed using LIBVISO2 [9], a freely available library for monocular and stereo visual odometry. To speed up inference, we subsample the data to a rate of one frame per second. Slower rates were found to suffer from excessive accumulated odometry error. For illustration purposes, here we extracted mid-size regions of OpenStreetMap data which included the true trajectory and the surrounding region. On average, they cover an area of 2km^2 and contain 47km of drivable roads. It is important to note that our method also localizes successfully on much larger maps, see Fig. 1 for example, which covers 18km^2 and contains 2,150km of drivable roads. We set the simplification threshold to $\epsilon = 10^{-2}$ which is applied when the number of mixture components for a segment is greater than one per 10m segment length.

Quantitative Evaluation: Quantitative results can be found in Table 1, with corresponding qualitative results shown in Fig. 8. Here, "M" and "S" indicate results using monocular and stereo visual odometry respectively. In addition, we computed odometry measurements from the GPS trajectories (entry "G" in the table) and ran our algorithm using the learned parameters from the stereo data. Note that this does not have access to absolute positions, but only relative position and orientation with respect to the previous frame. We also projected the GPS data onto the map data and measured the error produced by this projection. These errors, reported as "O" for oracle, are a lower bound on the best possible error to be achieved using the given map data. Note for some cases this error can be significant, as the map data does not account for lane widths, number of lanes or intersection sizes. Finally, we compute chance performance to be 397m by computing the average distance of the GPS data to the mean road position of each map.

We used the projected GPS data to learn the small number of model parameters. In particular, the street state evolution noise covariance $\Sigma_u^{\mathbf{x}}$, the angular AR(1) parameter γ_u and the observation noise $\Sigma_u^{\mathbf{y}}$ were estimated using maximum likelihood. We learn different parameters for highways and city/rural roads as the visual odometry performs significantly worse at higher speeds.

The accuracy of position and heading estimates is not well defined until the posterior has converged to a single mode. Thus, we only compute accuracy once a sequence has been *localized*. All results are divided into two temporally contiguous parts: unlocalized and localized. We define a sequence to be localized when for at least five seconds there is a single mode in the posterior and the distance to the ground truth position from that mode is less than 20 meters. Once the criteria for localization is met, all subsequent frames are considered localized. Errors in global position and heading of the MAP state for localized frames were computed using the GPS data as ground truth. Sequences which did not localize are indicated with a "*" in Table 1.

Overall, we are able to estimate the position and heading to 3.1m and 1.3° using stereo visual odometry. Note that this comes very close to the average oracle error of 1.44m, the lower bound on the achievable error induced by inaccuracies in the OpenStreetMap data! These results also outperform typical consumer grade navigation systems which offer accuracies of around 10 meters at best. Furthermore, errors are comparable to those achieved using the GPS-based odometry, suggesting the applicability and utility of low-cost vision-based sensors for localization. Using monocular odometry as input performs worse, but is still accurate to 18.4m and 3.6°, once it is localized. However, due to its stronger drift, it fails to localize in some cases as in sequence 01. This sequence contains highway driving only, where high speeds and sparse visual features make monocular visual odometry very challenging, leading to an accumulated error in the monocular odometry of more than 500m. In contrast, while the stereo visual odometry has somewhat higher than typical errors on this sequence, our method is still able to localize successfully as shown in Fig. 8.

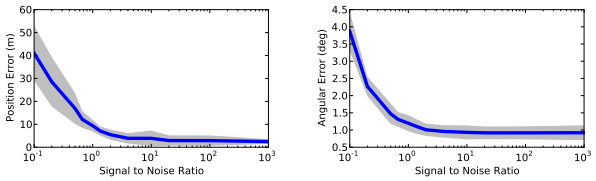

Figure 6. **Localization Accuracy with Noise:** Position and heading error with different noise levels. Averaged over five independent samples of noise.

Ambiguous Sequences: Sequences 04 and 06, shown in Fig. 7, are fundamentally ambiguous and cannot be localized with monocular, stereo or even GPS-based odometry. Sequence 04 is a short sequence on a straight road segment and, in the absence of any turns, cannot be localized beyond somewhere on the long road segment. Sequence 06 is longer and has turns, but traverses a symmetric path which results in a fundamental bimodality. In both cases our approach correctly indicates the set of probable locations.

Simplification Threshold: We study the impact of varying the mixture model simplification threshold. Fig. 4 depicts computation time per frame and localized position error averaged over sequences as a function of the threshold, ranging from 10^{-5} to 0.1 nats. We excluded sequences 04 and 06 as they are inherently ambiguous. As expected, computation time decreases and error increases with more simplification (*i.e.*, larger threshold). However, there is a point of diminishing returns for computation time around 10^{-2} nats, and little difference in error for smaller values. Thus we use a threshold of 10^{-2} for all other experiments.

Map Size: To investigate the impact of region size on localization performance, we assign uniform probability to portions of the map in a square region centered at the ground truth initial position and give zero initial probability to map locations outside the region. We varied the size of the square from 100m up to the typical map size of 2km, constituting an average of 300m to 47km of drivable road. We evaluated the time to localization for all non-ambiguous sequences (*i.e.*, all but 04, 06) and plotted the average as a function of the region size in Fig. 5. As expected, small initial regions allow for faster localization. Somewhat surprisingly, after the region becomes sufficiently large, the impact on localization becomes negligible. This is due to the inherent uniqueness of most sufficiently long paths, even in very large regions with many streets as the one shown in Fig. 1. While localization in a large and truly perfect Manhattan world with equiangular intersections and equilength streets would be nearly impossible based purely on odometry, such a world is not often realized as even Manhattan itself has non-perpendicular roads such as Broadway!

Noise: To study the impact of noise on the localization accuracy, we synthesized odometry measurements by adding

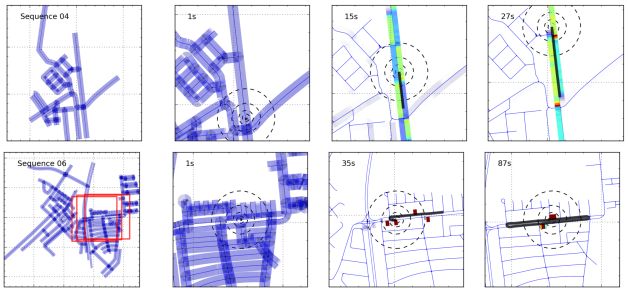

Figure 7. **Ambiguous Sequences:** Both 04 and 06 cannot be localized due to fundamental ambiguities. Sequence 04 consists of a short, straight driving sequence and 06 traverses a symmetric part of the map, resulting in two equally likely modes.

Gaussian noise to the GPS-based odometry. For each sequence five different samples of noisy odometry were created with signal-to-noise ratios (SNR) ranging from 0.1 to 1000. Fig. 6 depicts error in position and heading after localization. As expected, error increases as the SNR decreases, however the performance scales well, showing little change in error until the SNR drops below 1.

Scalability: Running on 16 cores with a basic Python implementation, we are able to achieve real time results as shown in Fig. 4 (right). To test the ability of our method to scale to large maps we ran the sequences using stereo odometry and a map covering the entire urban district of Karlsruhe, Germany. This map was approximately 18km^2 and had over 2,150km of drivable road. Despite this, the errors were the same as with the smaller maps and, while computation was slower, it still only took around 10 seconds per frame on average. We expect this could be greatly improved with suitable optimizations. Results on sequence 02 are shown in Fig. 1 and more are available in the supplemental material.

5. Conclusions

In this paper we have proposed an affordable approach to self-localization which employs (one or two) cameras mounted on the vehicle as well as crowd sourcing in the form of free online maps. We have demonstrated the effectiveness of our approach in a variety of diverse scenarios including highway, suburbs as well as crowded urban scenes. Furthermore, we have validated our approach on the KITTI visual odometry benchmark and shown that we are able to localize our vehicle with a precision of 3 m after only 20 seconds of driving. This is a new and exciting problem for computer vision and we believe there is much more to do. In particular, OpenStreetMaps contains many other salient pieces of information to aid in localization such as speed limits, street names, numbers of lanes, and more; we plan to exploit this information in the future. Finally, code and videos are available at `http://www.cs.toronto.edu/~mbrubake`.

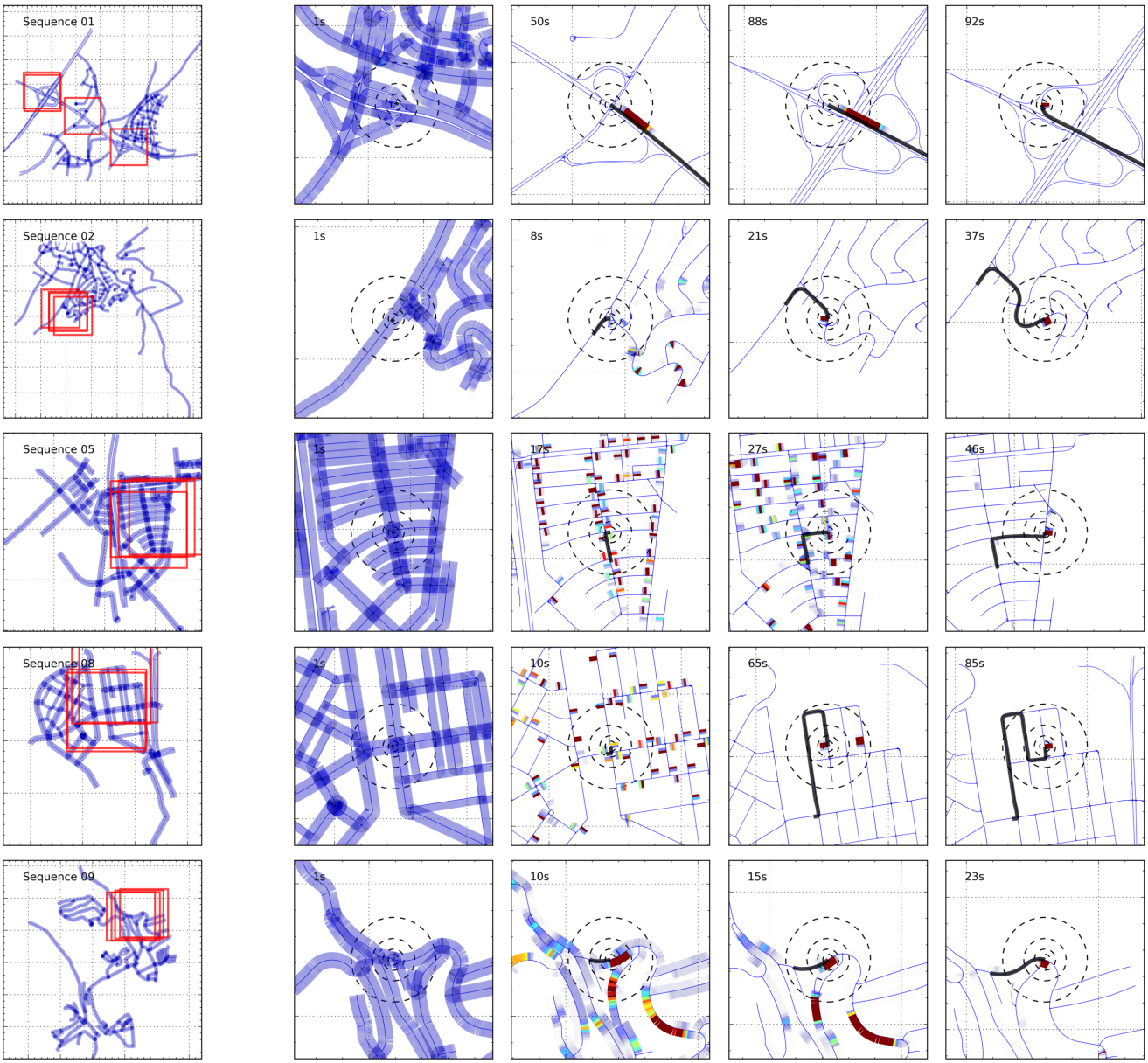

Figure 8. **Selected Frames:** Inference results for some of the sequences, full results can be found in the supplemental material. The left most column shows the full map region for each sequence, followed by zoomed in sections of the map showing the posterior distribution over time. The black line is the GPS trajectory and the concentric circles indicate the current GPS position. Grid lines are every 500m.

Acknowledgements Marcus A. Brubaker was supported by a Postdoctoral Fellowship from the Natural Sciences and Engineering Research Council of Canada.

References

[1] P. Alcantarilla, L. Bergasa, and F. Dellaert. Visual odometry priors for robust EKF-SLAM. In *ICRA*, 2010.

[2] G. Baatz, K. Köser, D. Chen, R. Grzeszczuk, and M. Pollefeys. Leveraging 3D City Models for Rotation Invariant Place-of-Interest Recognition. *IJCV*, 2012.

[3] H. Badino, D. Huber, and T. Kanade. Real-time topometric localization. In *ICRA*, May 2012.

[4] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *IJRR*, 27(6):647–665, 2008.

[5] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. *CVPR*, 1999.

[6] H. Durrant-Whyte and T. Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *RAM*, 2:2006, 2006.

[7] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *AAAI*, 1999.

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012.

[9] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *IV*, 2011.

[10] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *ICIRS*, 1998.

[11] J. Hays and A. A. Efros. im2gps: estimating geographic information from a single image. In *CVPR*, 2008.

[12] J. Hershey and P. Olsen. Approximating the Kullback-Leibler Divergence Between Gaussian Mixture Models. In *ICASSP*, volume 4, pages 317–320, 2007.

[13] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *IJRR*, 31:217–236, 2012.

[14] M. Kaess, K. Ni, and F. Dellaert. Flow separation for fast and robust stereo odometry. In *ICRA*, 2009.

[15] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *RSS*, 2007.

[16] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, 2012.

[17] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. Real: A system for large-scale mapping in constant-time using stereo. *IJCV*, pages 1–17, 2010.

[18] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *IJCAI*, 2003.

[19] D. Nister, O. Naroditsky, and J. R. Bergen. Visual odometry. In *CVPR*, 2004.

[20] S. M. Oh, S. Tariq, B. N. Walker, and F. Dellaert. Map-based priors for localization. In *ICIRS*, 2004.

[21] R. Paul and P. Newman. FAB-MAP 3D: Topological mapping with spatial and visual appearance. In *ICRA*, 2010.

[22] A. Pronobis, B. Caputo, P. Jensfelt, and H. Christensen. A discriminative approach to robust visual place recognition. In *IROS*, 2006.

[23] A. Rottmann, O. Martínez Mozos, C. Stachniss, and W. Burgard. Place classification of indoor environments with mobile robots using boosting. In *AAAI*, 2005.

[24] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *ICCV*, 2011.

[25] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *IJRR*, 21:735–758, 2002.

[26] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós. A comparison of loop closing techniques in monocular SLAM. *RAS*, 2009.

[27] J. Wu and J. M. Rehg. Where am I: Place instance and category recognition using spatial PACT. In *CVPR*, 2008.

A. Inference Details

To measure whether $f(\mathbf{s}_{t-1}) = p(u_t|\mathbf{x}_{t-1})$ is constant for a mixture component $\mathcal{N}(\mu^{(i)}, \Sigma^{(i)})$ we consider the function $g(\mu, \Sigma) = \int f(\mathbf{s}_{t-1})\mathcal{N}(\mathbf{s}_{t-1}|\mu, \Sigma)d\mathbf{s}_{t-1}$ which, in the case of the Gaussian CDF form of $f(\mathbf{s}_{t-1})$ can be shown to be a Gaussian CDF (proof in Supplementary Material). Dropping the index, i, if $\|\frac{d}{d\mu}g(\mu, \Sigma)\| < \eta$ for $\eta = 10^{-8}$ we consider $f(\mathbf{s}_{t-1})$ to be approximately constantly and the integral in Equation (10) can then be computed analytically as $f(\mu)\mathcal{N}(\mathbf{s}_t|\mathbf{A}\mu + \mathbf{b}, \Sigma^{\mathbf{x}} + \mathbf{A}\Sigma\mathbf{A}^T)$ which corresponds to the prediction step of a Kalman filter. If $\frac{d}{d\mu}g(\mu, \Sigma) \geq \eta$ then the mode overlaps the inflection point of $f(\mathbf{s}_{t-1})$ and the analytic model will not be a good approximation. Instead, we use a Monte Carlo approximation, drawing a set of $M = 400$ samples $\mathbf{s}_{t-1}^{(j)} \sim \mathcal{N}(\mu, \Sigma)$ for $j = 1, \ldots, M$ and approximate the integral with a single component $c\mathcal{N}(\mathbf{s}_t|\hat{\mu}, \hat{\Sigma})$ where $c = M^{-1}\sum_{j=1}^{M} f(\mathbf{s}_{t-1}^{(j)})$, and $\hat{\mu}, \hat{\Sigma}$ are found by moment matching to the Monte Carlo mixture approximation $\sum_{j=1}^{M} f(\mathbf{s}_{t-1}^{(j)})\mathcal{N}(\mathbf{s}_t|\mathbf{A}\mathbf{s}_{t-1}^{(j)} + \mathbf{b}, \Sigma^{\mathbf{x}})$. Once the integral in Equation (10) is approximated we must incorporate the observation $\mathbf{y}_t$. Because the observations are Gauss-Linear and the integral approximations are Gaussians this consists of multiplying two Gaussian distributions as in the update step of the Kalman filter.

Performing the above for each component and each pair of nodes produces a set of mixture model components for each u, the weights of which are proportional to P_u^t. After normalizing the mixtures for each street, normalizing across streets allows for the computation of P_u^t, the probability of being on a given street. The procedure for recursively updating the posterior is summarized in Algorithm 1 and more details can be found in the Supplemental Material.

B. Mixture Model Simplification

Given a Gaussian mixture model $f(x) = \sum_a \pi_a \mathcal{N}(x|\mu_a, \Sigma_a)$ we seek $g(x) = \sum_b \omega_b \mathcal{N}(x|\mu_b, \Sigma_b)$ with the least number of components such that $D(f\|g) < \epsilon$ where $D(f\|g)$ is the KL divergence. We begin with $g(x) = f(x)$ and successively remove the lowest weight component of $g(x)$ and update the remaining components to better fit $f(x)$ so long as $g(x)$ remains a good approximation. To compute the KL divergence $D(f\|g)$, we use instead a variational upper bound [12]. Introducing the variational parameters $\phi_{a,b} \geq 0$ and $\psi_{a,b} \geq 0$ such that $\sum_b \phi_{a,b} = \pi_a$ and $\sum_a \psi_{a,b} = \omega_b$, $D(f\|g) \leq \hat{D}(\phi, \psi, f, g)$ where $\hat{D}(\phi, \psi, f, g) = \sum_{a,b} \phi_{a,b}(\log \frac{\phi_{a,b}}{\psi_{a,b}} + D(f_a\|g_b))$ and $D(f_a\|g_b)$ is the KL divergence between $\mathcal{N}(x|\mu_a, \Sigma_a)$ and $\mathcal{N}(x|\mu_b, \Sigma_b)$. To compute the upper bound of $D(f\|g)$ we minimize $\hat{D}(\phi, \psi, f, g)$ with respect to the variational parameters ϕ and ψ. Similarly, to update the components of g we minimize $\hat{D}(\phi, \psi, f, g)$ with respect to the variational parameters ϕ and ψ as well as the parameters ω_b, μ_b and Σ_b. While this objective function is non-convex, for each set of parameters individually the exact minima can be found, providing an efficient coordinate-descent algorithm. The update equations for ϕ, ψ, ω_b, μ_b and Σ_b, along with the details of their derivation and a summary of the algorithm are found in the Supplementary Material.