Map-Based Probabilistic Visual Self-Localization

Marcus A. Brubaker, Andreas Geiger, and Raquel Urtasun,

Abstract—Accurate and efficient self-localization is a critical problem for autonomous systems. This paper describes an affordable solution to vehicle self-localization which uses odometry computed from two video cameras and road maps as the sole inputs. The core of the method is a probabilistic model for which an efficient approximate inference algorithm is derived. The inference algorithm is able to utilize distributed computation in order to meet the real-time requirements of autonomous systems in some instances. Because of the probabilistic nature of the model the method is capable of coping with various sources of uncertainty including noise in the 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, the proposed method is able to localize a vehicle to 4m on average after 52 seconds of driving on maps which contain more than 2,150km of drivable roads.

Index Terms—localization, visual odometry, OpenStreetMaps, map

1 Introduction

One of the fundamental tasks necessary for autonomous systems is to be able to accurately determine its position at all times. This is necessary not only for core tasks like path planning [1] and navigation [2], but also to simplify other tasks such as scene understanding [3], [4], [5]. Despite decades of research, self-localization is still an exciting open problem. In this paper we are interested in building *affordable* and *robust* solutions to this problem in the context of autonomous driving.

Currently, the most common self-localization technology is the Global Positioning System (GPS) which utilizes satellites in medium Earth orbit to provide location information. While popular due to its low cost, GPS has limitations which make it potentially unsuitable as a stand-alone technology for autonomous systems. Notably, its signal is not always available and its localization can become imprecise, *e.g.*, in the presence of skyscrapers, tunnels or jammed signals. While this may not be problematic when helping a human navigate, it can have catastrophic consequences for autonomous devices like self-driving cars. Further, in applications involving underground or indoor navigation where GPS is completely unavailable an alternative approach is required.

Several alternatives have been developed in the past few years, which frame self-localization as a retrieval task. Towards this goal, the "world" is represented in terms of visual features [6], [7], [8], [9], [10], [11] or 3D point clouds [12], [13], [14], [15], and localization is performed by retrieving images or point clouds which are similar to the current scene. In combination with GPS, impressive results have been demonstrated, *e.g.*, at the DARPA Urban Challenge [16] or the Google self-driving car which has autonomously driven over 500,000 km without incident as of August 2012[1]. However it remains unclear if maintaining an up-to-date world representation will be feasible given the computation, memory and communication requirements. Furthermore, privacy issues need to be considered as recording and storing such data might be illegal in some countries.

All aforementioned approaches have in common that they require significant knowledge of the appearance of the world for localization to be possible. In contrast, in this paper we propose an approach which is able to localize a vehicle in places that have never before been seen. We take our inspiration from humans, which are able to perform this task while having only access to a rough cartographic description of the environment.

In particular, we propose to exploit OpenStreetMap (OSM), a free online community-driven map, which is free and constantly updated, making our approach inexpensive. Towards this goal, we derive a probabilistic map localization approach that uses visual odometry estimates and OSM data as only inputs. We demonstrate the effectiveness of our approach on the KITTI visual odometry benchmark sequences [17]. As our experiments show, we are able to self-localize our vehicle after 52 seconds of driving with an accuracy of 4 meters on a 18km^2 map containing $2,150$km of drivable roads. A preliminary version of this work was presented in [18].

2 Related Work

The problem of *map localization* (sometimes known as the *kidnapped robot problem*) has been traditionally approached using Monte Carlo methods [12], [13], [14], [15]. Here, the Markov assumption is used to maintain a particle-based posterior representation of an agent's pose, integrating laser or visual observations. Typically, these methods operate only in a small environment without providing any global (geographic) positioning information. Furthermore, and most significantly, they rely on more specific measurement sources (*e.g.*, depth measurements, sonar) and are restricted to small-scale environments (*e.g.*, office scenes) where accurate 2D floor plans are available. While the approach presented here is similar to these

• *This work was supported by a postdoctoral fellowship from the Natural Science and Engineering Research Council of Canada (NSERC).*

1. http://googleblog.blogspot.ca/2012/08/the-self-driving-car-logs-more-miles-on.html

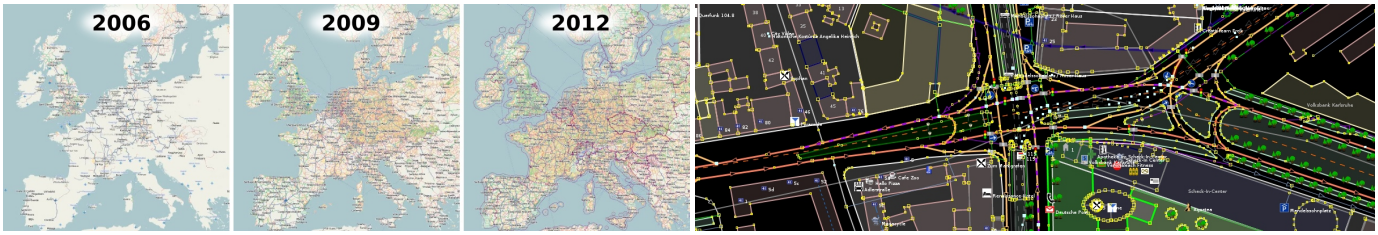

Fig. 1. **The OpenStreetMap project: Free geographic data for the world.** Left: Development and coverage of OpenStreetMap from 2006-2012. Right: Screenshot of JOSM, a popular OSM editor, showing the level of annotation detail for the intersection 'Mendelssohnplatz' in Karlsruhe, Germany.

methods in that a recursive Bayesian filtering algorithm is proposed, these methods use particle-based filtering techniques which do not handle persistent uncertainty due to the well-known issue of particle depletion. In contrast, the proposed approach leverages analytic approximations, making it more stable when ambiguities persist over long periods (*e.g.*, see Figure 5). Maps have been widely used to improve upon GPS measurements as well as to fill in when signals are unavailable or degraded [19], [20], [21], [22]. However, these methods assume that a relatively accurate initial position has been provided, *e.g.*, from a recently lost GPS signal. In contrast, this work assumes no knowledge of the position of the vehicle beyond the gross region.

In computer vision *place recognition* methods attempt to localize [6], [8], [10], [11], [23], [24], [25] or categorize [26], [27], [28] an image, given a large database of geo-referenced images. Often, hashing methods in combination with a geometric verification step are employed to register the camera view to a 3D model of the scene which has been obtained a-priori. 3D input information is used in [29], where regression forests are applied to localize an RGB-D camera within an indoor scene. When video streams or time stamped image sequences are available, temporal dependencies can be modelled as well [7], [9], [30]. While some of the proposed methods have been applied successfully to single landmarks or even at city-scale, they either can't provide the required accuracy [8], [31] or they require a large amount of reference images for building the models. However, creating an up-to-date "world database" seems at best costly if not impractical. While services such as Google Street View have been successful at capturing images for many locations, much of the world has still not been captured and maintaining updated images will likely remain a challenge. Furthermore, vision-based place recognition methods must be robust against changes in lighting across day times and seasons [32] to achieve reliable and fail-safe localization, a requirement hard to fulfill in practice. In contrast, the maps used by our localization approach take up only a couple of gigabytes for storing maps for the entirety of planet earth[2] and can be updated relatively quickly and inexpensively.

When one assumes that the initial position is known, *visual odometry* [33] yields relative motion estimates that can be integrated to obtain an estimate for the agent's current location. While impressive performance on the KITTI visual odometry benchmark [17] has been demonstrated for example in [34], [35], [36], the incremental nature of these methods inevitably leads to drift, limiting the error of the leading methods at around 1% in terms of translation and 0.003 deg/m in terms of rotation.

Simultaneous Localization And Mapping (SLAM) methods [37], [38], [39], [40], [41], [42], [43] are able to reduce this drift by using landmarks and jointly optimizing over all or a selection of poses and landmarks. Efficient optimization strategies using incremental sparse matrix factorization [44] or relative representations [45] have been proposed to make these algorithms tractable and scalable. Due to the lack of texture in indoor scenes, RGB-D approaches leveraging Microsoft's Kinect sensor have been proposed recently [46], [47], [48], [49], [50], [51]. Drift can be further reduced by revisiting places for several times and detecting loop closures in the traveled trajectory [52], [53], [54], [55]. Incorporating these constraints into the optimization leads to improved maps and reduces the localization error. Unfortunately, SLAM methods can only localize in maps that have been recorded a-priori with a sensor setup similar to the one used at localization time and their performance strongly depends on the frequency with which places are revisited. In contrast, the proposed approach enables geographic localization without knowledge of the initial location, relies only on freely available OpenStreetMap information and doesn't assume loopy trajectories.

The task of localization in road network maps has also been approached using so-called *map matching* techniques [56], [57], [58], [59], [60], [61], [62]. With a goal similar to ours, these methods are able to efficiently localize a query map within a larger map region. While in principle such query maps can be obtained by integrating visual odometry measurements over time [63], neglecting drift can heavily impact performance. A notable exception is [64] which aims at localizing non-rigid subgraphs in a larger graph. However, their algorithm applies only to small graphs with up to 100 nodes. By proposing a probabilistic map localization model that explicitly models visual odometry measurements and their noise characteristics we mitigate these problems and obtain fast localization times even on very large road maps with more than $2,000$km of drivable roads.

2. http://wiki.openstreetmap.org/wiki/planet.osm

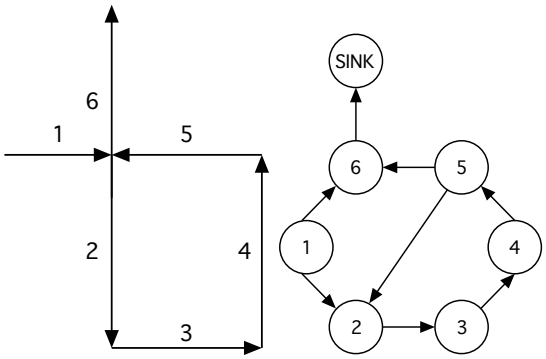

Fig. 2. **Map Representation:** This figure shows a simple street map (left) and its corresponding graph representation (right). Directed street segments are represented as nodes and edges define their connectivity.

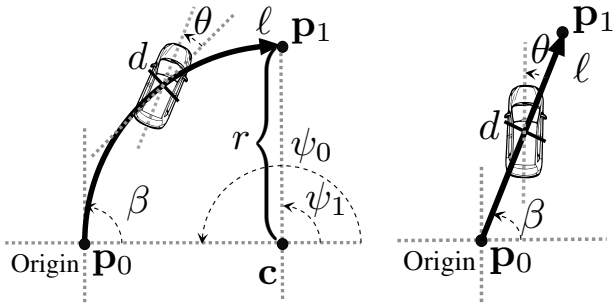

Fig. 3. **Street Segment:** Each street segment has a start and end position $\mathbf{p}_0$ and $\mathbf{p}_1$, a length ℓ, an initial heading of the street segment β and a curvature parameter $\alpha = \frac{\psi_1 - \psi_0}{\ell}$. For arc segments $\mathbf{c}$ is the circle center, r is the radius and ψ_0 and ψ_1 are the start and end angles of the arc. For linear segments, $\alpha = 0$.

3 Visual Localization

The goal of this paper is to use one or two roof-mounted cameras as well as a cartographic map of the environment to self-localize a driving vehicle. The map contains streets as line segments as well as intersection points, but no visual appearance information. As such, the only information that can be used for self-localization is the trajectory of the vehicle and as such we propose the use of visual odometry. Note that other sources of information such as the speedometer can also be used, as they provide similar information. As the trajectory resulting from the integration of visual odometry is too noisy for direct shape matching in large maps due to drift, 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. We define a graph-based representation of the map as well as a probabilistic model of how a vehicle can traverse the graph. Furthermore, we derive a filtering algorithm to perform inference in the probabilistic model, which exploits the structure of the graph and properties of the model to efficiently perform approximate inference. In order to keep running times reasonable, we propose to manage the complexity of the mixture models using a novel algorithm for simplifying Gaussian Mixture models which are used to represent the posterior distribution. In the remainder of the section, we first discuss the employed map information, followed by our probabilistic model and the inference algorithm.

3.1 The OpenStreetMap Project

In 2004, Steve Coast started the OpenStreetMap project with the goal of creating a free editable map of the world much like Wikipedia provides a free community-based encyclopedia. While initial efforts focused mainly on mapping the United Kingdom, volunteers around the globe quickly helped in making OSM a worldwide success. Users can contribute to OSM in various ways, for example by supplying GPS tracks using portable GPS devices (*e.g.*, while driving, cycling or hiking), labeling objects such as buildings in aerial imagery or by providing local information. So far, more than 1.5 million[3] users have registered and provided geographic information to OSM. As of December 2013, more than 3.7 billion GPS track points have been submitted corresponding to over 2.1 billion waypoints and 210 million roads in total.

The growth of OSM for the region of Europe is illustrated in Figure 1 (left). Compared to commercial products like Google Maps, the provided data is more up-to-date, includes many details and – most importantly – can be freely accessed and used under the Open Database License. The startup GeoFabrik[4], for example, provides daily snapshots for certain regions or the whole globe (29 GB) which can be downloaded in XML format. Figure 1 (right) shows a screenshot of JOSM[5], one of the most popular OSM editors. Note that the annotations do not only include streets, but also cycling and hiking trails, traffic lights, signs, parking lots, building outlines, shops, trees, postboxes, bars, restaurants, cinemas, recreational areas and more. While the main focus of this work is on leveraging road maps for localization, we believe that the wealth of OSM data will become useful in other areas of computer vision, such as urban scene understanding [3] or semantic image segmentation of road scenes [65].

For our purposes, we extract all crossings and all drivable roads (represented as piece-wise linear segments) connecting them. We additionally extract the type of each street (*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 the state of the vehicle is defined.

3.2 Map Representation

The map itself is represented by a directed graph where nodes represent street segments and edges define the connectivity of those segments. Without loss of generality, all street segments are assumed to be one-way and two-way streets are converted to two one-way streets to fit this framework. Roads which

3. http://wiki.openstreetmap.org/wiki/stats
4. http://www.geofabrik.de/
5. http://josm.openstreetmap.de/

dead-end or run off the edge of the map are connected to a "sink" node. Figure 2 represents an example of a map and the corresponding graph representation.

We find it convenient to represent every street segment as either a linear or a circular arc segment, which can both be described using the same parametric representation as detailed below. Note that more complex road shapes can be approximated with these primitives by joining multiple, short segments. As illustrated in Figure 3, we define the shape of a street segment by its starting and ending points $\mathbf{p}_0$ and $\mathbf{p}_1$ respectively, initial heading angle β and a curvature parameter α. The curvature is zero for linear segments (*i.e.*, $\alpha = 0$) and $\alpha = \frac{\psi_1 - \psi_0}{\ell}$ for circular arc segments where ψ_0 and ψ_1 are the starting and ending angle of the arc and ℓ is the arc length of the segment.

The position and orientation of a vehicle restricted to a location on the map is then defined by the street segment u that the vehicle is driving on, the distance from the origin of that street segment d and the angular offset θ of the vehicle heading from the local street heading. Thus, the global heading of the vehicle is given by

$$\bar{\theta} = \theta + \beta + \alpha d \tag{1}$$

and its position is

$$\bar{\mathbf{p}} = \frac{\ell - d}{\ell}\mathbf{p}_0 + \frac{d}{\ell}\mathbf{p}_1 \tag{2}$$

for a linear segment and

$$\bar{\mathbf{p}} = \mathbf{c} + r\,\mathbf{d}\left(\frac{\ell - d}{\ell}\psi_0 + \frac{d}{\ell}\psi_1\right) \tag{3}$$

for a circular arc segment where r is the radius of the circle, $\mathbf{c}$ denotes the circle center and $\mathbf{d}(\psi) = (\cos\psi, \sin\psi)^T$. We refer the reader to Figure 3 for an illustration of all parameters of the street segment geometry. Important to note is that this parameterization of heading allows the rate of global heading change $\partial\bar{\theta}$ to be dependent on the linear velocity $\partial d/\partial t$ while still allowing the dynamics to be modelled with a simple linear diffusion model described below.

Given this map representation, we next present a state-space model which describes the the motion of the vehicle in the map. A probabilistic inference algorithm is then derived to recursively compute the posterior over position. This filtering algorithm is summarized in Algorithm 1.

3.3 State-Space Model

As visual odometry measurements are obtained at discrete time intervals, we choose a discrete time model where the subscript t represents the index of a data timestamp or – equivalently – the frame number. However, for brevity, we will call t simply 'time' in the following. Based on the map representation introduced above, the state of the vehicle at time t is described by $\mathbf{x}_t = (u_t, \mathbf{s}_t)$ 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 and the hat indicates that both are defined relative to the current street u_t which changes when the vehicle transitions to a new street segment. The visual odometry measurement computed at time t is denoted by $\mathbf{y}_t$ and directly measures the linear and angular displacement from time $t-1$ to time t. We assume that the map data available is planar so visual odometry measurements are projected onto the road plane which can be easily recovered using SfM or stereo information. While height information is sometimes available in OSM, it is typically noisy and unreliable so it was not used. Note, however, that the model presented here could be extended to the 3D case should reliable height data be available. The odometry observations are modelled as

$$p(\mathbf{y}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{y}_t|\mathbf{M}_{u_t}\mathbf{s}_t, \Sigma^{\mathbf{y}}_{u_t}) \tag{4}$$

where $\mathbf{M}_u = [\mathbf{m}_d, \mathbf{m}_\theta]^T$, $\mathbf{m}_d = (1, -1, 0, 0)^T$ and $\mathbf{m}_\theta = (\alpha_u, -\alpha_u, 1, -1)^T$. Note that the curvature of the street, α_u, is necessary because the global heading of the vehicle, $\bar{\theta}$, depends on both d and θ, see (1). The state transition model is factorized as

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = p(u_t|\mathbf{x}_{t-1})p(\mathbf{s}_t|u_t, \mathbf{x}_{t-1}) \tag{5}$$

with street transition probability $p(u_t|\mathbf{x}_{t-1})$, and state transition distribution $p(\mathbf{s}_t|u_t, \mathbf{x}_{t-1})$. The latter is assumed to be the result of a linear transformation corrupted by Gaussian noise

$$p(\mathbf{s}_t|u_t, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{s}_t|\mathbf{A}_{u_t,u_{t-1}}\mathbf{s}_{t-1} + \mathbf{b}_{u_t,u_{t-1}}, \Sigma^{\mathbf{s}}_{u_t}) \tag{6}$$

with $\Sigma^{\mathbf{s}}_{u_t}$ the covariance matrix for a given u_t which is learned from data as discussed in Section 4. A second-order, constant velocity model is used to represent the change in d

$$d_t = d_{t-1} + (d_{t-1} - \hat{d}_{t-2}) \tag{7}$$

and a first order autoregressive model, *i.e.*, AR(1), for the angular offset θ

$$\theta_t = \gamma_{u_{t-1}}\theta_{t-1} \tag{8}$$

Algorithm 1 Filter

Input: Posterior at $t-1$, $\{P^{t-1}_u, \mathcal{M}^{t-1}_u\}$
Input: Observation at t, $\mathbf{y}_t$
Initialize mixtures, $\mathcal{M}^t_u \leftarrow \varnothing$, for all u
for all streets u_{t-1} **do**
  for all streets u_t reachable from u_{t-1} **do**
    $\mathcal{M}' \leftarrow \varnothing$
    for all $(\omega, \mu, \Sigma) \in \mathcal{M}^{t-1}_{u_{t-1}}$ **do**
      Compute $c_{pred}\mathcal{N}(\mu_{pred}, \Sigma_{pred})$ using Alg 2
      Compute $c_{upd}\mathcal{N}(\mu_{upd}, \Sigma_{upd})$ using Alg 3
      $\mathcal{M}' \leftarrow \mathcal{M}' \bigcup \{(c_{upd}, \mu_{upd}, \Sigma_{upd})\}$
    if $u_t \neq u_{t-1}$ **then**
      Compute (c, μ, Σ) to approximate $\mathcal{M}'$
      $\mathcal{M}^t_{u_t} \leftarrow \mathcal{M}^t_{u_t} \bigcup \{(c, \mu, \Sigma)\}$
    else
      $\mathcal{M}^t_{u_t} \leftarrow \mathcal{M}^t_{u_t} \bigcup \mathcal{M}'$
for all streets u **do**
  $P^t_u \leftarrow \sum_{(c,\mu,\Sigma)\in\mathcal{M}^t_u} c$
  $\mathcal{M}^t_u \leftarrow \{(\frac{c}{P^t_u}, \mu, \Sigma) \mid (c, \mu, \Sigma) \in \mathcal{M}^t_u\}$
  if $\frac{\ell_u}{|\mathcal{M}^t_u|} < 10$ meters **then**
    Simplify $\mathcal{M}^t_u$ with Algorithm 4
Normalize P^t_u so that $\sum_u P^t_u = 1$.
For all u, if $P^t_u < 10^{-50}$ set $P^t_u \leftarrow 0$ and $\mathcal{M}^t_u \leftarrow \varnothing$
Return: Posterior at t, $\{P^t_u, \mathcal{M}^t_u\}$

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}. Even though these models are relatively simple, they were found to be very effective in practice. Combining (6), (7) and (8), one obtains

$$\mathbf{A}_{u_t,u_{t-1}} = \begin{bmatrix} 2 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \gamma_{u_t} & 0 \\ 0 & \alpha_{u_{t-1}} - \alpha_{u_t} & 1 & 0 \end{bmatrix} \tag{9}$$

$$\mathbf{b}_{u_t,u_{t-1}} = \begin{cases} -(\ell_{u_{t-1}}, \ell_{u_{t-1}}, 0, \theta_{u_t,u_{t-1}})^T & u_t \neq u_{t-1} \\ (0,0,0,0)^T & u_t = u_{t-1} \end{cases} \tag{10}$$

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}.

As noted above and illustrated in Figure 3, 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 $\mathbf{s}_t$ is relative to u_t. The length of the u_{t-1} must be subtracted from both, and $\hat{\theta}_{t-1}$ must 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 street transition probability $p(u_t|\mathbf{x}_{t-1})$ defines the probability of transitioning onto street u_t given the previous state $\mathbf{x}_{t-1}$. To define the probability of changing street segments we make use of the Gaussian state transition model in (6) to get

$$p(u_t|\mathbf{x}_{t-1}) \propto \xi_{u_t,u_{t-1}} \int_{\ell_{u_{t-1}}}^{\ell_{u_{t-1}}+\ell_u} \mathcal{N}(x|\mathbf{a}_d^T\mathbf{s}_{t-1}, \mathbf{a}_d^T\Sigma^{\mathbf{s}}_{u_{t-1}}\mathbf{a}_d)dx \tag{11}$$

where $\mathbf{a}_d = (2,-1,0,0)$ is the first row of $\mathbf{A}_{u_t,u_{t-1}}$ which represents the predicted position according to (7),

$$\xi_{u_t,u_{t-1}} = \begin{cases} 1 & u_t = u_{t-1} \\ \frac{1}{|\mathbf{N}(u_{t-1})|} & u_t \in \mathbf{N}(u_{t-1}) \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

is the prior probability of transitioning onto u_t from u_{t-1}, and $\mathbf{N}(u)$ is the set of streets to which u connects.

Extremely short street segment can be problematic, as they must be skipped entirely when a vehicle is traveling quickly enough. To handle this, we add "leapfrog" edges to the graph which allow the vehicle to move from u_{t-1} to any u_t to which there exists a path in the base graph. As the speed of the vehicle is assumed to be limited, not all possible edges need to be added. Instead, only edges representing paths of 30m or less are added which handles vehicles traveling up to approximately 110km/h with observations every second. If faster travel is necessary or observations are less frequent, then longer edges can easily be added. To incorporate "leapfrog" edges into the model, the entries of $\mathbf{b}_{u_t,u_{t-1}}$ are updated to consider also transitioning over multiple street segments and $\xi_{u_t,u_{t-1}}$ becomes the product of all ξ along the path.

3.4 Inference

Given the model described above, inference consists of computing the filtering distribution, $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. Like most recursive filtering algorithms (*e.g.*, the particle or Kalman filters) we will divide the algorithm into a prediction step where uncertainty is propagated using the state transition model (5) and an update step where the uncertainty is updated using the observation model (4). The overall inference algorithm is described in Algorithm 1 and the prediction and update steps are derived next.

Algorithm 2 Prediction Step

Input: Parameters of current mode μ, Σ
Input: Street nodes u_t, u_{t-1}
if $\|\frac{d}{d\mu}g(\mu,\Sigma)\| < \eta$ **then**
  Analytically approximate $c_{pred}\mathcal{N}(\mu_{pred}, \Sigma_{pred})$
  $c_{pred} \leftarrow p(u_t|u_{t-1}, \mathbf{s}_{t-1} = \mu)$
  $\mu_{pred} \leftarrow \mathbf{A}_{u_t,u_{t-1}}\mu + \mathbf{b}_{u_t,u_{t-1}}$
  $\Sigma_{pred} \leftarrow \Sigma^{\mathbf{s}}_{u_t} + \mathbf{A}_{u_t,u_{t-1}}\Sigma\mathbf{A}^T_{u_t,u_{t-1}}$
else
  Sample to compute $c_{pred}\mathcal{N}(\mu_{pred}, \Sigma_{pred})$
  for $j = 1, \ldots, M$ **do**
    $\mathbf{s}^{(j)}_{t-1} \sim \mathcal{N}(\mu, \Sigma)$
    $\mathbf{s}^{(j)}_t \leftarrow \mathbf{A}_{u_t,u_{t-1}}\mathbf{s}^{(j)}_{t-1} + \mathbf{b}_{u_t,u_{t-1}}$
    $w^{(j)} \leftarrow p(u_t|u_{t-1}, \mathbf{s}_{t-1} = \mathbf{s}^{(j)}_{t-1})$
  $c_{pred} \leftarrow M^{-1}\sum_{j=1}^M w^{(j)}$
  $\mu_{pred} \leftarrow (Mc_{pred})^{-1}\sum_{j=1}^M w^{(j)}\mathbf{s}^{(j)}_t$
  $\Sigma_{pred} \leftarrow \Sigma^{\mathbf{s}}_{u_t} + \sum_{j=1}^M w^{(j)}\frac{(\mathbf{s}^{(j)}_t - \mu_{pred})(\mathbf{s}^{(j)}_t - \mu_{pred})^T}{Mc_{pred}}$
Return: Predicted mode $(c_{pred}, \mu_{pred}, \Sigma_{pred})$

Remembering that $\mathbf{x}_t = (u_t, \mathbf{s}_t)$, the posterior can be factored using the product rule as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = p(\mathbf{s}_t|u_t, \mathbf{y}_{1:t})p(u_t|\mathbf{y}_{1:t}) \tag{13}$$

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. The discrete distribution $p(u_t|\mathbf{y}_{1:t})$ is easily represented as a multinomial distribution over street labels. For representing the continuous distribution $p(\mathbf{s}_t|u_t, \mathbf{y}_{1:t})$ a Gaussian mixture model is used, *i.e.*,

$$p(\mathbf{s}_t|u_t, \mathbf{y}_{1:t}) = \sum_{i=1}^{N_{u_t}} \pi^{(i)}_{u_t}\mathcal{N}(\mathbf{s}_t|\mu^{(i)}_{u_t}, \Sigma^{(i)}_{u_t}) \tag{14}$$

where N_{u_t} is the number of components for the mixture associated with u_t and $\mathcal{M}^t_{u_t} = \{\pi^{(i)}_{u_t}, \mu^{(i)}_{u_t}, \Sigma^{(i)}_{u_t}\}_{i=1}^{N_{u_t}}$ are the parameters of the mixture for u_t. This is a general and powerful representation and allows for efficient and accurate inference.

Assuming independent observations given the states and first order Markov state transitions, the filtering distribution can then be written recursively as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \int \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1} \tag{15}$$

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

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \sum_{u_{t-1}} \frac{P_{u_{t-1}}}{Z_t} p(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{s}_t|u_t, u_{t-1}, \mathbf{s}_{t-1}) \times p(u_t|u_{t-1}, \mathbf{s}_{t-1})p(\mathbf{s}_{t-1}|u_{t-1}, \mathbf{y}_{1:t-1})d\mathbf{s}_{t-1} \tag{16}$$

Algorithm 3 Update Step

Input: Current mode $(c_{pred}, \mu_{pred}, \Sigma_{pred})$ on street u_t
Input: Observation $\mathbf{y}_t$
$\Sigma_{upd} \leftarrow \left(\mathbf{M}_{u_t}^T {\Sigma_{u_t}^{\mathbf{y}}}^{-1} \mathbf{M}_{u_t} + \Sigma_{pred}^{-1}\right)^{-1}$
$\mu_{upd} \leftarrow \Sigma_{upd}\left(\mathbf{M}_{u_t}^T {\Sigma_{u_t}^{\mathbf{y}}}^{-1} \mathbf{y}_t + \Sigma_{pred}^{-1}\mu_{pred}\right)$
$\Sigma'_{upd} \leftarrow \Sigma_{u_t}^{\mathbf{y}} + \mathbf{M}_{u_t}\Sigma_{pred}\mathbf{M}_{u_t}^T$
$c_{upd} \leftarrow \frac{\omega c_{pred}|\Sigma_{upd}|^{0.5}}{|\Sigma_{pred}|^{0.5}|\Sigma_{u_t}^{\mathbf{y}}|^{0.5}} \exp\left(-\frac{1}{2}\|\mathbf{y}_t - \mathbf{M}_{u_t}\mu_{pred}\|^2_{\Sigma'_{upd}}\right)$
Return: Updated mode $(c_{upd}, \mu_{upd}, \Sigma_{upd})$.

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 the mixture model (14) and the state transition dynamics (6) into (16), the integrand becomes

$$\sum_{i=1}^{N_{u_{t-1}}} \pi^{(i)} \int p(u_t|u_{t-1},\mathbf{s}_{t-1})\mathcal{N}(\mathbf{s}_t|\mathbf{A}\mathbf{s}_{t-1}+\mathbf{b},\Sigma^{\mathbf{s}}) \times \mathcal{N}(\mathbf{s}_{t-1}|\mu^{(i)},\Sigma^{(i)})d\mathbf{s}_{t-1} \quad (17)$$

where the subscripts have been dropped for clarity. In general, the integral in (17) is not analytically tractable due to the non-linear form of $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$. However, if $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$ was constant the integral could be solved easily. In the model presented above $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$ from (11) is the Gaussian CDF and has sigmoidal shape (see Figure 4). Consequently, it is approximately constant everywhere except near the transition point of the sigmoid. In the following this is exploited in order to create an efficient but accurate approximate inference algorithm.

3.4.1 Prediction Step

For each mode i representing a distribution over $\mathbf{s}_{t-1}$ we aim to determine whether $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$ can be considered constant and whether an analytic approximation can be used. Looking at Figure 4, the goal is to determine whether the mode, $\mathcal{N}(\mathbf{s}_{t-1}|\mu^{(i)}, \Sigma^{(i)})$ lies in either tail (*i.e.*, the blue portion) or if it is close to the changepoint of the sigmoid.

To that end, consider the function

$$g(\mu, \Sigma) = \int p(u_t|u_{t-1}, \mathbf{s}_{t-1})\mathcal{N}(\mathbf{s}_{t-1}|\mu, \Sigma)d\mathbf{s}_{t-1} \quad (18)$$

and the norm of its gradient with respect to μ, *i.e.*, $\|\frac{d}{d\mu}g(\mu, \Sigma)\|$ where we dropped the mode index i for notational clarity. Substituting (11) in for $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$, and simplifying gives

$$g(\mu, \Sigma) = \xi_{u_t,u_{t-1}}\left(\Phi(\ell_{u_{t-1}} + \ell_u|m, s^2) - \Phi(\ell_{u_{t-1}}|m, s^2)\right) \quad (19)$$

where $m = \mathbf{a}_d^T\mu$, $s^2 = \mathbf{a}_d^T(\Sigma^{\mathbf{s}}_{u_{t-1}} + \Sigma)\mathbf{a}_d$ and $\Phi(\cdot|m, s^2)$ is the CDF of a univariate normal distribution with mean m and variance s^2.

Using this simplification, there are now two cases based on $\|\frac{d}{d\mu}g(\mu, \Sigma)\|$. If $\|\frac{d}{d\mu}g(\mu, \Sigma)\| < \eta$ for some threshold[6], then $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$ is considered to be approximately constant

6. In practice, we used $\eta = 10^{-8}$.

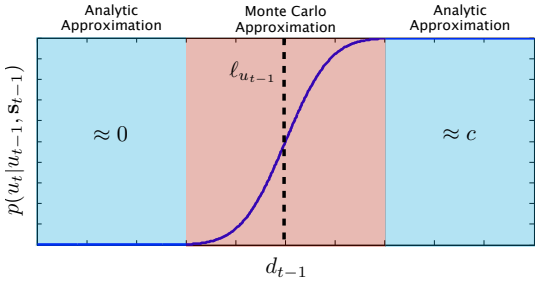

Fig. 4. **Approximate Inference:** The sigmoidal shape of the street node transition probability allows for efficient inference as analytic approximations are effective in the tails and a Monte Carlo approximation can be applied for the central section which is very small in practice.

with respect to the given mode and the integral in (17) is approximated analytically as

$$c_{pred}\,\mathcal{N}(\mathbf{s}_t|\mathbf{A}\mu + \mathbf{b}, \Sigma^{\mathbf{s}} + \mathbf{A}\Sigma\mathbf{A}^T) \quad (20)$$

where $c_{pred} = p(u_t|u_{t-1}, \mu)$. This update is equivalent to the prediction step of a Kalman filter. Note that most streets we consider are relatively long, thus the section where $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$ can be considered constant occurs frequently and we benefit from the analytical approximation. Furthermore, in those cases when $c_{pred} \approx 0$ (see left side of Figure 4) the mode can simply be dropped.

If $\|\frac{d}{d\mu}g(\mu, \Sigma)\| \geq \eta$ then the mode overlaps the inflection point of $p(u_t|u_{t-1}, \mathbf{s}_{t-1})$ and the analytic approximation will not suffice. 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 the integral is then approximated with a single component $c_{pred}\,\mathcal{N}(\mathbf{s}_t|\mu_{pred}, \Sigma_{pred})$ where $c_{pred} = \frac{1}{M}\sum_{j=1}^{M} w^{(j)}$, and

$$\mu_{pred} = \frac{1}{Mc_{pred}}\sum_{j=1}^{M} w^{(j)}\mathbf{s}_t^{(j)} \quad (21)$$

$$\Sigma_{pred} = \Sigma^{\mathbf{s}} + \frac{1}{Mc_{pred}}\sum_{j=1}^{M} w^{(j)}(\mathbf{s}_t^{(j)} - \mu_{pred})(\mathbf{s}_t^{(j)} - \mu_{pred})^T$$

are found by matching moments to the Monte Carlo approximation

$$\sum_{j=1}^{M} p(u_t|u_{t-1}, \mathbf{s}_{t-1}^{(j)})\mathcal{N}(\mathbf{s}_t|\mathbf{A}\mathbf{s}_{t-1}^{(j)} + \mathbf{b}, \Sigma^{\mathbf{s}}) \quad (22)$$

with $\mathbf{s}_t^{(j)} = A\mathbf{s}_{t-1}^{(j)} + b$ and $w^{(j)} = p(u_t|u_{t-1}, \mathbf{s}_{t-1} = \mathbf{s}_{t-1}^{(j)})$. The prediction step is summarized in Algorithm 2.

3.4.2 Update Step

The result of the prediction step is a single mode $c_{pred}\,\mathcal{N}(\mathbf{s}_t|\mu_{pred}, \Sigma_{pred})$ which approximates the integral in (17). This mode must then be updated according to (16) in order to take account for observation $\mathbf{y}_t$. Because the observation distribution $p(\mathbf{y}_t|\mathbf{x}_t)$ is Gaussian and linear in $\mathbf{s}_t$, this corresponds to multiplying two Gaussian densities and can

Algorithm 4 GMM Simplification

Input: Mixture model parameters $\mathcal{M}$
Initialize $\mathcal{M}' = \mathcal{M}$
loop
  Select a component to remove $\hat{b} = \arg\min_{b'} \omega_{b'}$
  $\hat{\mathcal{M}} \leftarrow \mathcal{M}' \setminus \{(\omega_{\hat{b}}, \mu_{\hat{b}}, \Sigma_{\hat{b}})\}$
  Initialize the variational parameters ϕ and ψ
  while $\hat{D}(\phi, \psi, \mathcal{M}, \hat{\mathcal{M}}) \geq \epsilon$ and not converged **do**
    Minimize $\hat{D}(\phi, \psi, \mathcal{M}, \hat{\mathcal{M}})$ using (29-33)
  if $\hat{D}(\phi', \psi', \mathcal{M}, \hat{\mathcal{M}}) \geq \epsilon$ **then**
    Return: $\mathcal{M}'$
  else
    $\mathcal{M}' \leftarrow \hat{\mathcal{M}}$

be solved in closed form. The resulting update can be written as $c_{upd}\mathcal{N}(\mathbf{s}_t|\mu_{upd}, \Sigma_{upd})$ where

$$c_{upd} = \frac{\omega c_{pred}|\Sigma_{upd}|^{0.5}}{|\Sigma_{pred}|^{0.5}|\Sigma^{\mathbf{y}}_{u_t}|^{0.5}} e^{-\frac{1}{2}\|\mathbf{y}_t - \mathbf{M}_{u_t}\mu_{pred}\|^2_{\Sigma'_{upd}}} \quad (23)$$

$$\Sigma_{upd} = \left(\mathbf{M}^T_{u_t} {\Sigma^{\mathbf{y}}_{u_t}}^{-1} \mathbf{M}_{u_t} + \Sigma^{-1}_{pred}\right)^{-1} \quad (24)$$

$$\mu_{upd} = \Sigma_{upd}\left(\mathbf{M}^T_{u_t} {\Sigma^{\mathbf{y}}_{u_t}}^{-1} \mathbf{y}_t + \Sigma^{-1}_{pred}\mu_{pred}\right) \quad (25)$$

with $\Sigma'_{upd} = \Sigma^{\mathbf{y}}_{u_t} + \mathbf{M}_{u_t}\Sigma_{pred}\mathbf{M}^T_{u_t}$. We summarize this update step in Algorithm 3.

Performing the prediction and update steps for each component and each pair of connected nodes produces a set of mixture model components for each u, the weights of which are proportional to P^t_u. After normalizing the mixtures for each street, normalizing across streets yields P^t_u, the probability of being located on a given street. The filtering process is summarized in Algorithm 1. Note that this algorithm can be performed in parallel by assigning subsets of streets to different workers, a fact which we exploit to achieve real-time performance.

3.5 Managing Posterior Complexity

The previous section provides a basic algorithm to perform inference by computing the filtering distribution recursively. Unfortunately, straightforward application of this algorithm is intractable as the complexity of the posterior (*i.e.*, the number of mixture components) grows exponentially with time as modes are duplicated at transitions with multiple options (*e.g.*, intersections). This can be seen by noting the two loops in Algorithm 1 over u_t and u_{t-1}. This means that every mode in the current posterior can, in the next posterior, result in multiple modes, yielding a worst case exponential growth. To alleviate this, three approximations are used which limit the resulting complexity of the posterior. These approximations have been found 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 when $u_{t-1} \neq u_t$ are all likely similar. As such, all of the transitioned modes are replaced with a single component using moment matching. Second, after running the algorithm for a while most streets will have negligible probability but may still be associated with mixtures with a large number of components. To avoid this, mixtures on streets whose probability $p(u_t|\mathbf{y}_{1:t})$ is below a threshold are pruned such that these streets will have zero probability. To avoid mistakes, we make use of a very conservative threshold, $p(u_t|\mathbf{y}_{1:t}) \leq 10^{-50}$, in our experiments. Finally, even with the above steps, the number of components in the posterior grows with t. Many of the components will have small weight and be redundant. To mitigate this, a mixture model simplification procedure is run when the number of modes on a street segment gets too large. This procedure, which we describe next, removes components and updates others while keeping the KL divergence below a specified threshold ϵ. The value for ϵ was determined experimentally, as described in Section 4.4.

3.5.1 Mixture Model Simplification

Given a Gaussian mixture model with $N = |\mathcal{M}|$ components and parameters denoted by $\mathcal{M} = \{(\pi_a, \mu_a, \Sigma_a)\}_{a=1}^N$ and probability density function

$$f(x) = \sum_a \pi_a \mathcal{N}(x|\mu_a, \Sigma_a) \quad (26)$$

we seek a mixture model $\mathcal{M}' = \{(\omega_b, \mu_b, \Sigma_b)\}_{b=1}^{N'}$ with probability density function $g(x) = \sum_b \omega_b \mathcal{N}(x|\mu_b, \Sigma_b)$ which has the least number of components (*i.e.*, minimal $N' = |\mathcal{M}'|$) such that $D(f\|g) < \epsilon$ where $D(f\|g)$ is the KL divergence. We begin with $\mathcal{M}' = \mathcal{M}$ and successively remove the component with the lowest weight from $\mathcal{M}'$. At the same time, we update the remaining components to minimize the KL divergence $D(f\|g)$. This process, of removing components and updating the remaining ones, continues so long as $\mathcal{M}'$ remains a good approximation in the sense that $D(f\|g) < \epsilon$.

To minimize the KL divergence $D(f\|g)$, we use a variational upper bound [66]. 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 $\sum_a \psi_{a,b} = \omega_b$, we can bound the KL divergence by

$$D(f\|g) \leq \hat{D}(\phi, \psi, \mathcal{M}, \mathcal{M}') \quad (27)$$

where

$$\hat{D}(\phi, \psi, \mathcal{M}, \mathcal{M}') = \sum_{a,b} \phi_{a,b}\left(\log\frac{\phi_{a,b}}{\psi_{a,b}} + D(f_a\|g_b)\right) \quad (28)$$

and $D(f_a\|g_b)$ is the KL divergence between the Gaussian distributions $\mathcal{N}(x|\mu_a, \Sigma_a)$ and $\mathcal{N}(x|\mu_b, \Sigma_b)$. In order to simply compute the upper bound of $D(f\|g)$, $\hat{D}(\phi, \psi, \mathcal{M}, \mathcal{M}')$ is minimized with respect to the variational parameters ϕ and ψ.

Similarly, to update the components of $\mathcal{M}'$ to better approximate $\mathcal{M}$, $\hat{D}(\phi, \psi, \mathcal{M}, \mathcal{M}')$ is minimized with respect to both the variational parameters ϕ and ψ and the component parameters ω_b, μ_b and Σ_b. While this objective function is non-convex, the exact minima for each set of parameters can be found conditioned on the others. This provides the basis for an efficient coordinate-descent algorithm which, while not globally optimal, was found to be effective in practice. Using the method of Lagrange multipliers to enforce the constraints

TABLE 1

Quantitative Evaluation: Average position and heading error and driving time until localization. "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. Stereo and GPS-based odometry results run on small maps as well as the full region map. Position and heading errors are computed over frames which have been localized. Sequences 04 and 06 are ambiguous and cannot be localized, see text and Figure 5.

		00	01	02	03	05	07	08	09	10	Average
Position	M	16 ± 67m	893 ± 22m	8.1 ± 7.6m	19 ± 7m	5.6 ± 4.6m	15 ± 15m	45 ± 106m	5.4 ± 4.7m	534 ± 25m	39 ± 135m
	S	2.1 ± 1.5m	5.1 ± 6.7m	4.1 ± 2.9m	4.8 ± 2.0m	2.6 ± 1.4m	1.8 ± 1.1m	6.0 ± 21.4m	4.2 ± 3.1m	3.9 ± 1.9m	3.7 ± 9.5m
	G	1.8 ± 1.1m	3.6 ± 5.0m	2.2 ± 1.3m	6.9 ± 2.9m	2.7 ± 1.6m	1.5 ± 0.8m	2.4 ± 7.4m	3.8 ± 3.0m	2.5 ± 1.6m	2.4 ± 3.6m
	S (full)	2.2 ± 1.8m	2.6 ± 2.2m	4.3 ± 3.0m	6.8 ± 1.6m	3.7 ± 2.4m	1.7 ± 1.0m	6.0 ± 21.4m	4.4 ± 3.3m	4.9 ± 2.0m	4.0 ± 9.8m
	G (full)	1.9 ± 1.1m	3.8 ± 4.8m	2.1 ± 1.3m	8.4 ± 2.3m	4.6 ± 3.3m	1.5 ± 0.8m	1.9 ± 1.1m	4.0 ± 3.1m	2.8 ± 1.3m	2.6 ± 2.2m
	O	0.8 ± 0.6m	1.3 ± 1.2m	1.0 ± 0.8m	2.5 ± 1.8m	1.3 ± 1.1m	0.6 ± 0.4m	1.1 ± 0.9m	1.2 ± 1.2m	1.1 ± 0.8m	1.2 ± 1.0m
Heading	M	$2.0 \pm 3.9°$	$5.2 \pm 3.5°$	$1.5 \pm 1.2°$	$2.4 \pm 1.6°$	$2.0 \pm 1.8°$	$1.3 \pm 1.0°$	$10 \pm 30°$	$1.6 \pm 1.4°$	$116 \pm 8°$	$5.4 \pm 20.3°$
	S	$1.2 \pm 1.3°$	$2.7 \pm 1.8°$	$1.3 \pm 1.0°$	$1.6 \pm 1.2°$	$1.4 \pm 1.2°$	$1.9 \pm 1.3°$	$1.2 \pm 1.5°$	$1.3 \pm 1.1°$	$1.3 \pm 1.2°$	$1.3 \pm 1.3°$
	G	$1.0 \pm 1.0°$	$0.9 \pm 0.8°$	$0.8 \pm 0.9°$	$1.4 \pm 0.9°$	$1.2 \pm 1.1°$	$1.5 \pm 1.1°$	$1.0 \pm 1.3°$	$0.9 \pm 0.8°$	$1.0 \pm 1.0°$	$1.0 \pm 1.1°$
	S (full)	$1.3 \pm 1.2°$	$1.4 \pm 0.9°$	$1.2 \pm 1.1°$	$1.2 \pm 0.6°$	$1.3 \pm 1.1°$	$2.3 \pm 1.5°$	$1.2 \pm 1.6°$	$1.3 \pm 1.4°$	$1.4 \pm 1.2°$	$1.3 \pm 1.3°$
	G (full)	$1.0 \pm 1.0°$	$1.3 \pm 0.8°$	$0.9 \pm 0.9°$	$1.4 \pm 0.6°$	$1.2 \pm 1.1°$	$1.3 \pm 1.0°$	$1.0 \pm 1.3°$	$1.0 \pm 0.9°$	$1.0 \pm 0.9°$	$1.0 \pm 1.1°$
Driving Time	M	22s	88s	26s	41s	199s	34s	79s	25s	94s	68 ± 54s
	S	22s	88s	26s	34s	45s	26s	70s	24s	18s	39 ± 23s
	G	21s	88s	26s	34s	44s	26s	78s	25s	18s	40 ± 24s
	S (full)	41s	90s	27s	66s	63s	34s	70s	44s	32s	52 ± 20s
	G (full)	41s	88s	27s	64s	55s	32s	79s	44s	32s	51 ± 21s

$\sum_b \omega_b = 1$, $\sum_b \phi_{a,b} = \pi_a$ and $\sum_a \psi_{a,b} = \omega_b$, the minima of (28) with respect to the individual parameters are achieved by

$$\omega_b = \sum_a \phi_{a,b} \tag{29}$$

$$\psi_{a,b} = \omega_b \frac{\phi_{a,b}}{\sum_{a'} \phi_{a',b}} \tag{30}$$

$$\phi_{a,b} = \pi_a \frac{\psi_{a,b} \exp\left(-D(f_a \| g_b)\right)}{\sum_{b'} \psi_{a,b'} \exp\left(-D(f_a \| g_{b'})\right)} \tag{31}$$

$$\mu_b = \frac{\sum_a \phi_{a,b} \mu_a}{\sum_a \phi_{a,b}} \tag{32}$$

$$\Sigma_b = \frac{\sum_a \phi_{a,b} \left(\Sigma_a + (\mu_a - \mu_b)(\mu_a - \mu_b)^T\right)}{\sum_a \phi_{a,b}} \tag{33}$$

These equations are used iteratively to simplify the mixture model and this procedure is summarized in Algorithm 4. The variational parameters ϕ and ψ are initialized based on the weights of the mixture before components are removed and, after an individual component is removed, it's ϕ and ψ values are uniformly distributed to the remaining components.

4 Experimental Evaluation

To evaluate the localization method in realistic situations, experiments were performed using the KITTI visual odometry dataset [17]. The 11 training sequences were used for quantitative evaluation as ground truth data[7] is available. The dataset consists of two video streams from a left and right camera. Visual odometry was computed from these videos using LIBVISO2 [35], a freely available library for monocular and stereo visual odometry. To reduce computational time, the visual odometry is subsampled to a rate of one frame per second. Slower data rates are possible and may be faster but were found to suffer from accumulated odometry error. To better illustrate the results, mid-size regions of OpenStreetMap data were extracted for each sequence which included the true trajectory and the surrounding region. On average, these subregions covered an area of 2km^2 and contain 47km of drivable roads. Unless otherwise specified, all experiments were initialized with regularly placed mixture components which approximated a uniform probability of being at any point on the map. The method is also able to localize successfully on much larger maps, see results in Figures 11 and 12, where the map covers an 18km^2 region and contains 2,150km of drivable roads.

7. An OxTS RT 3003 GPS unit was used which integrates GPS, RTK corrections, IMU, and wheel odometry and has a rated accuracy of 0.02m.

4.1 Quantitative Evaluation

Quantitative results can be found in Table 1, with corresponding qualitative results shown in Figures 9 and 10. Here, "M" and "S" indicate results using monocular and stereo visual odometry respectively with localization performed on the subregion maps. In addition, "gold-standard" odometry measurements were computed based on the GPS trajectories (entry "G" in the table) and the algorithm was run using the parameters learned from the stereo data. Note that this does not use the absolute positions from the GPS signal, only relative position and orientation with respect to the previous frame. Results on the full 18km^2 map are indicated with "(full)". Finally, the method was also run using KITTI test visual odometry sequences. These do not have publicly available ground-truth so cannot be quantitatively assessed but results can be viewed online: http://www.cs.toronto.edu/~mbrubake.

To determine the accuracy of the maps and GPS data, the ground truth GPS tracks were projected onto the nearest street segment on the map and the position reprojection error was computed. These errors, reported as "O" for oracle, are indicative of the best possible error which can be achieved using the given map data. In some cases this error can be significant, as the map data does not account for lane

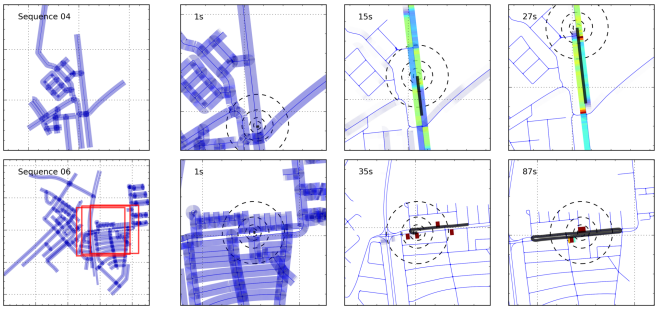

Fig. 5. **Ambiguous Sequences:** Both 04 and 06 cannot be localized due to fundamental ambiguities. Sequence 04 consists of a short, straight driving sequence and 06 traverses a symmetric part of the map, resulting in two equally likely modes.

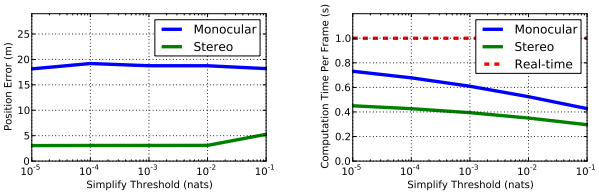

Fig. 6. **Simplification Threshold:** Impact of the simplification threshold ϵ on localization accuracy (left) and computation time (right). See Section 4.4.

widths, number of lanes or intersection sizes. Finally, chance performance was computed to be 397m by measuring the average distance of the GPS data to the mean road position of each subregion map.

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

The accuracy of position and heading estimates is not well defined until the posterior has converged to a single mode. Thus, accuracy is only computed once a sequence has been *localized*. All results are divided into two temporally contiguous parts: unlocalized and localized. A sequence is defined to be *localized* when, for at least ten seconds, there is a single mode in the posterior. 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.

Overall, the position and heading were estimated to an accuracy of 3.7m and 1.3° using stereo visual odometry on the subregion maps. Note that this is within the standard deviation of the oracle error of 2.6 ± 2.2m, which is a lower bound on the achievable error induced by inaccuracies in the map and GPS data. These results outperform typical consumer grade navigation systems which offer accuracy of around 10m at best and is better than the 5.19m reported by [63] on

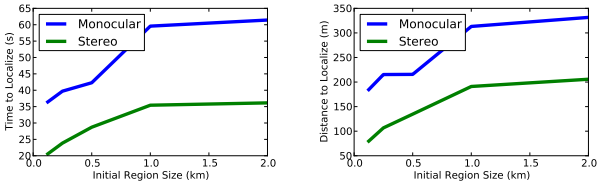

Fig. 7. **Map Size:** Driving time (left) and distance travelled (right) before localization as a function of the map size.

the same sequences. 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 and suffers catastrophic failures on sequences 01, 08 and 10. However, excluding these sequences, it was accurate to 11.5m. Sequence 01 is particularly challenging for the monocular odometry as it is exclusively highway driving where high speeds and sparse visual features results in an accumulated odometry errors of more than 500m, making accurate localization impossible. Sequence 08 localizes well initially, but a large outlier in the monocular odometry after localization results in significant errors at the end of the sequence. In contrast the stereo visual odometry performs well on all sequences shown in Figures 9 and 10 and is able to localize successfully in all but sequences 04 and 06 which are discussed next.

4.2 Identifiable Sequences

A natural question to ask is what makes localization of a vehicle difficult. Part of this answer comes from two sequences which the system was unable to localize, sequences 04 and 06. As shown in Figure 5, these sequences 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.

More generally though, one can look at how long unambiguous sequences took to localize. Consider the two sequences which localized most quickly on the full maps, sequences 02 and 10. Looking at these sequences in Figures 11 and 12, shows that both sequences traversed curved roads, making them quickly uniquely identifiable. In contrast, consider the two sequences which took the longest to localize on the full maps, sequences 01 and 08. These sequences demonstrate that long straight stretches are difficult to localize and even with turns, as in sequence 08, can be problematic. However, also interesting to note is that both of these sequences showed no practical difference between the subregion and full map localization time.

4.3 Simplification Threshold

To determine the value used for the simplification threshold ϵ multiple values were tried. Figure 6 depicts the average

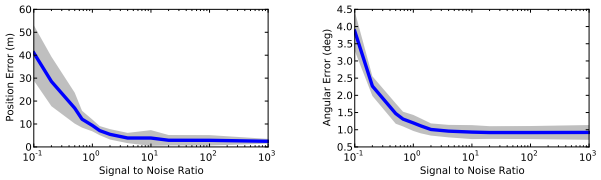

Fig. 8. **Localization Accuracy with Noise:** Position and heading error with different noise levels, averaged over five independent samples of noise.

computation time per frame and localized position error as a function of the threshold, with values ranging from 10^{-5} to 0.1 nats. Sequences 04 and 06 were excluded from this experiment because, as discussed above, 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 a simplification threshold of $\epsilon = 10^{-2}$ was selected and used for all other experiments.

4.4 Map Size

To understand how the map region size impacts performance, experiments were run where uniform initial probability was restricted to a square subregion of the map centred at the ground truth initial position and all other parts of the map were given zero initial probability. The size of the square was varied from 100m up to the typical subregion map size of 2km, resulting in an average of 300m to 47km of drivable road. For this range of sizes, the average time to localization was computed for all non-ambiguous sequences (*i.e.*, all but 04, 06) and was plotted as a function of region size in Figure 7. As expected, small initial regions localized faster. However, somewhat surprisingly, after the region becomes sufficiently large, the impact of region size on localization diminishes. Indeed, average localization time (right column of Table 1) on the full maps was only 13 seconds longer on average than on the subregion maps. This suggests that most paths quickly become unique as they get longer, even in very large regions.

4.5 Noise

To study the impact of noise on the localization accuracy, odometry measurements were synthesized by adding Gaussian noise to the GPS-based odometry. Five different samples of noisy odometry were created for each sequence with signal-to-noise ratios (SNR) ranging from 0.1 to 1000. Error in position and heading after localization is plotted in Figure 8 as a function of SNR. As expected, error increases as the SNR decreases, however the performance scales well, showing little change in error until the SNR drops below 1.

4.6 Scalability

Running on 16 cores with a basic Python implementation, we are able to achieve real time results on the subregion maps as shown in Figure 6 (right). To test the methods ability to scale to large maps we ran the sequences using stereo odometry with a map covering the entire urban district of Karlsruhe, Germany. The map is around 18km^2 and contains over $2,150$km of drivable road. Errors and localization times are shown in Table 1 for both stereo and GPS-based odometry, indicated by "S (full)" and "G (full)" respectively. The errors are effectively equivalent to those with the smaller maps and the localization time is only slightly longer. Computation was slower, taking around 20 seconds per frame on average. However, it is expected that this could be greatly improved with suitable optimization and further parallelization.

5 Conclusions

In this paper we have described an 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. The effectiveness of our approach was demonstrated in a variety of scenarios including highway, suburbs and crowded urban scenes. Furthermore, the approach has been validated on the KITTI visual odometry benchmark and shown to be able to localize the vehicle with an average precision of 4m after less than a minute of driving. This is a new and exciting problem for computer vision and we believe there is much more to do. Experiments with monocular odometry suggest that noise in the observations is likely heavy tailed, as there are periodic, large outliers, and extending the model to handle this would improve its robustness. The probabilistic framework here could be easily extended to incorporate other visual cues of location, for instance, learning to classify street types from video frames. In particular, OpenStreetMaps contains other salient pieces of information to aid in localization such as speed limits, street names, and more; we plan to exploit this information in the future. Code and videos are available at http://www.cs.toronto.edu/~mbrubake.

References

[1] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *IJRR*, vol. 29, no. 5, pp. 485–501, 2010.
[2] D. Delling, P. Sanders, D. Schultes, and D. Wagner, "Engineering route planning algorithms," in *Algorithmics of large and complex networks*. Springer, 2009, pp. 117–139.
[3] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3d traffic scene understanding from movable platforms," *PAMI*, 2014.
[4] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. Torr, "Semantic modelling of urban scenes," in *ICRA*, 2013.
[5] D. Munoz, J. A. Bagnell, and M. Hebert, "Stacked hierarchical labeling," in *ECCV*, 2010.
[6] G. Baatz, K. Köser, D. Chen, R. Grzeszczuk, and M. Pollefeys, "Leveraging 3D City Models for Rotation Invariant Place-of-Interest Recognition," *IJCV*, 2012.
[7] H. Badino, D. Huber, and T. Kanade, "Real-time topometric localization," in *ICRA*, May 2012.
[8] J. Hays and A. A. Efros, "im2gps: estimating geographic information from a single image," in *CVPR*, 2008.
[9] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," in *RSS*, 2007.
[10] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, "Worldwide pose estimation using 3d point clouds," in *ECCV*, 2012.
[11] T. Sattler, B. Leibe, and L. Kobbelt, "Fast image-based localization using direct 2d-to-3d matching," in *ICCV*, 2011.
[12] F. Dellaert, W. Burgard, D. Fox, and S. Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization," *CVPR*, 1999.

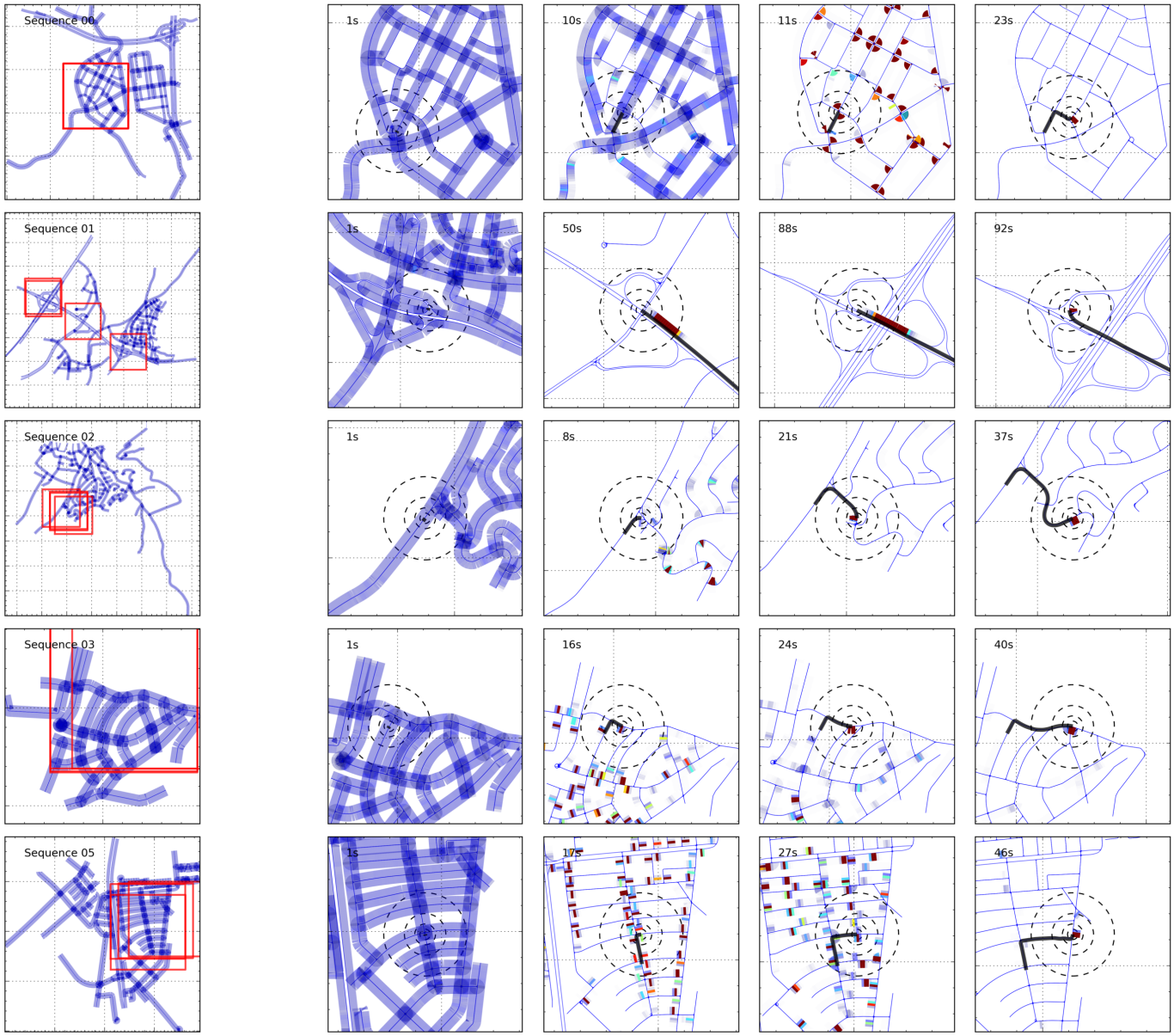

Fig. 9. **Selected Frames on the Subregion Maps:** Inference results for unambiguous sequences. The left most column shows the full map for each sequence, followed by zoomed in sections of the map showing the posterior distribution over time. The black line is the GPS trajectory and the concentric circles indicate the current GPS position. Grid lines are every 500m. High probability is indicated with red while low probability regions are shown in blue.

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

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

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

[16] M. Buehler, K. Iagnemma, and S. Singh, Eds., *The DARPA Urban Challenge*, ser. Advanced Robotics, vol. 56, 2009.

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

[18] M. A. Brubaker, A. Geiger, and R. Urtasun, "Lost! Leveraging the Crowd for Probabilistic Visual Self-Localization," in *CVPR*, 2013.

[19] M. E. El Najjar and P. Bonnifait, "A road-matching method for precise vehicle localization using belief theory and kalman filtering," *Autonomous Robots*, vol. 19, no. 2, pp. 173–191, 2005.

[20] J. Guivant and R. Katz, "Global urban localization based on road maps," in *IROS*, 2007, pp. 1079–1084.

[21] S. Bonnabel and E. Salaün, "Design and prototyping of a low-cost vehicle localization system with guaranteed convergence properties," *Control Engineering Practice*, vol. 19, no. 6, pp. 591–601, 2011.

[22] C. Fouque and P. Bonnifait, "Matching raw gps measurements on a navigable map without computing a global position," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 887–898, 2012.

[23] F. Li and J. Kosecka, "Probabilistic location recognition using reduced feature set," in *ICRA*, 2006.

[24] W. Zhang and J. Kosecka, "Image based localization in urban environments," in *3DPVT*, 2006.

[25] G. Schindler, M. Brown, and R. Szeliski, "City-scale location recognition," in *CVPR*, 2007.

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

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

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

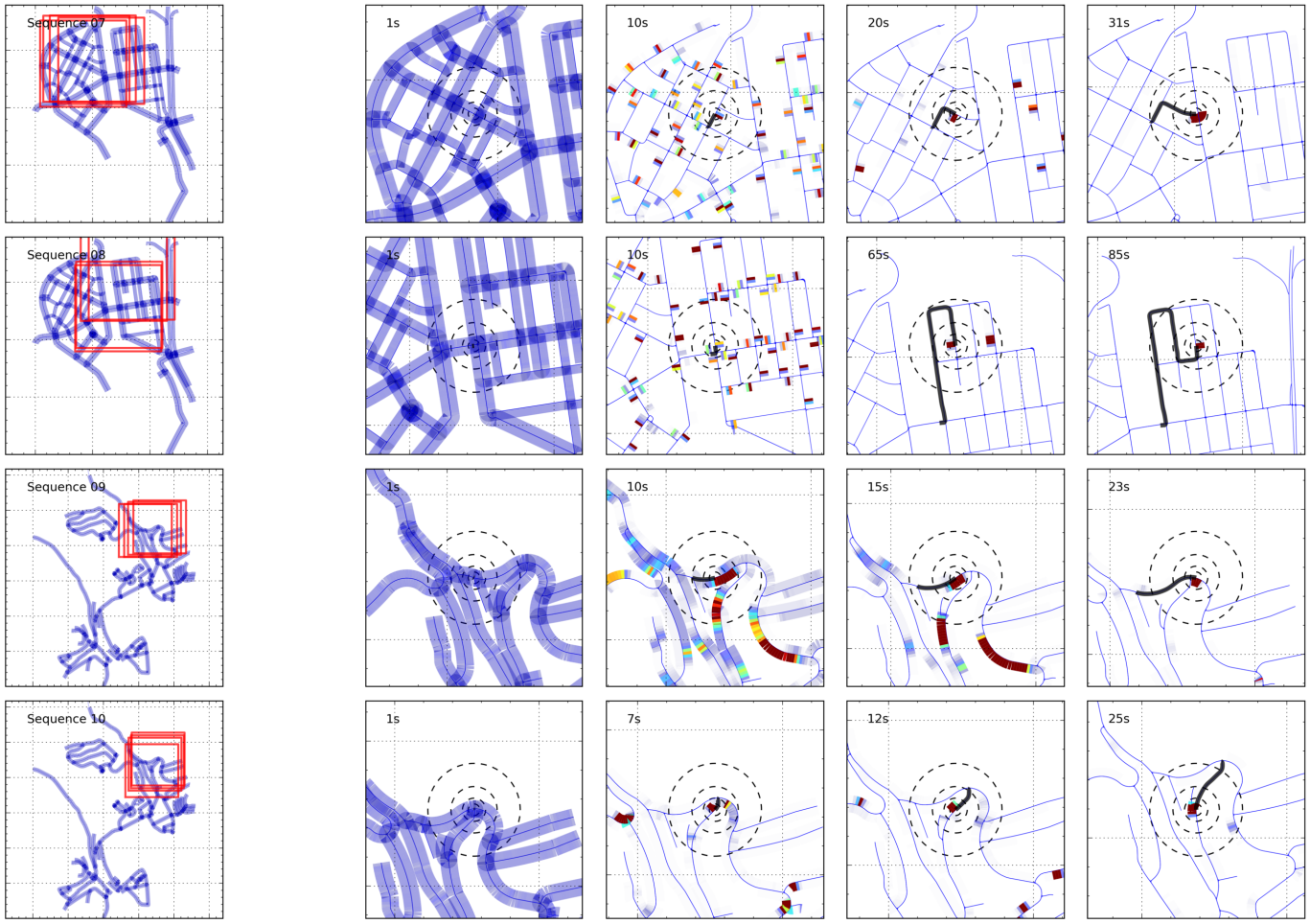

Fig. 10. **Selected Frames on the Subregion Maps:** Inference results for unambiguous sequences. The left most column shows the full map for each sequence, followed by zoomed in sections of the map showing the posterior distribution over time. The black line is the GPS trajectory and the concentric circles indicate the current GPS position. Grid lines are every 500m. High probability is indicated with red while low probability regions are shown in blue.

[29] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *CVPR*, 2013.

[30] E. Kalogerakis, O. Vesselova, J. Hays, A. A. Efros, and A. Hertzmann, "Image sequence geolocation with human travel priors," in *ICCV*, 2009.

[31] R. Dewri, P. Annadata, W. Eltarjaman, and R. Thurimella, "Inferring trip destinations from driving habits data," in *WPES*, 2013.

[32] W. S. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *IJRR*, 2013.

[33] D. Nister, O. Naroditsky, and J. R. Bergen, "Visual odometry," in *CVPR*, 2004.

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

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

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

[37] 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.

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

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

[40] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera slam," *PAMI*, vol. 29, 2007.

[41] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time." in *CVPR*, 2011.

[42] A. Ranganathan and F. Dellaert, "Online probabilistic topological mapping," *IJRR*, vol. 30, no. 6, pp. 755–771, 2011.

[43] A. Ranganathan, E. Menegatti, and F. Dellaert, "Bayesian inference in the space of topological maps," *Transactions on Robotics*, 2006.

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

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

[46] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *ICCV*, 2013.

[47] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, "Real-time camera tracking and 3d reconstruction using signed distance functions," in *RSS*, 2013.

[48] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *IROS*, 2013.

[49] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *ISMAR*, 2011.

[50] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the rgb-d slam system," in *ICRA*, 2012.

[51] T. Whelan, M. Kaess, J. Leonard, and J. McDonald, "Deformation-based loop closure for large scale dense RGB-D SLAM," in *IROS*, Tokyo, Japan, November 2013.

[52] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization

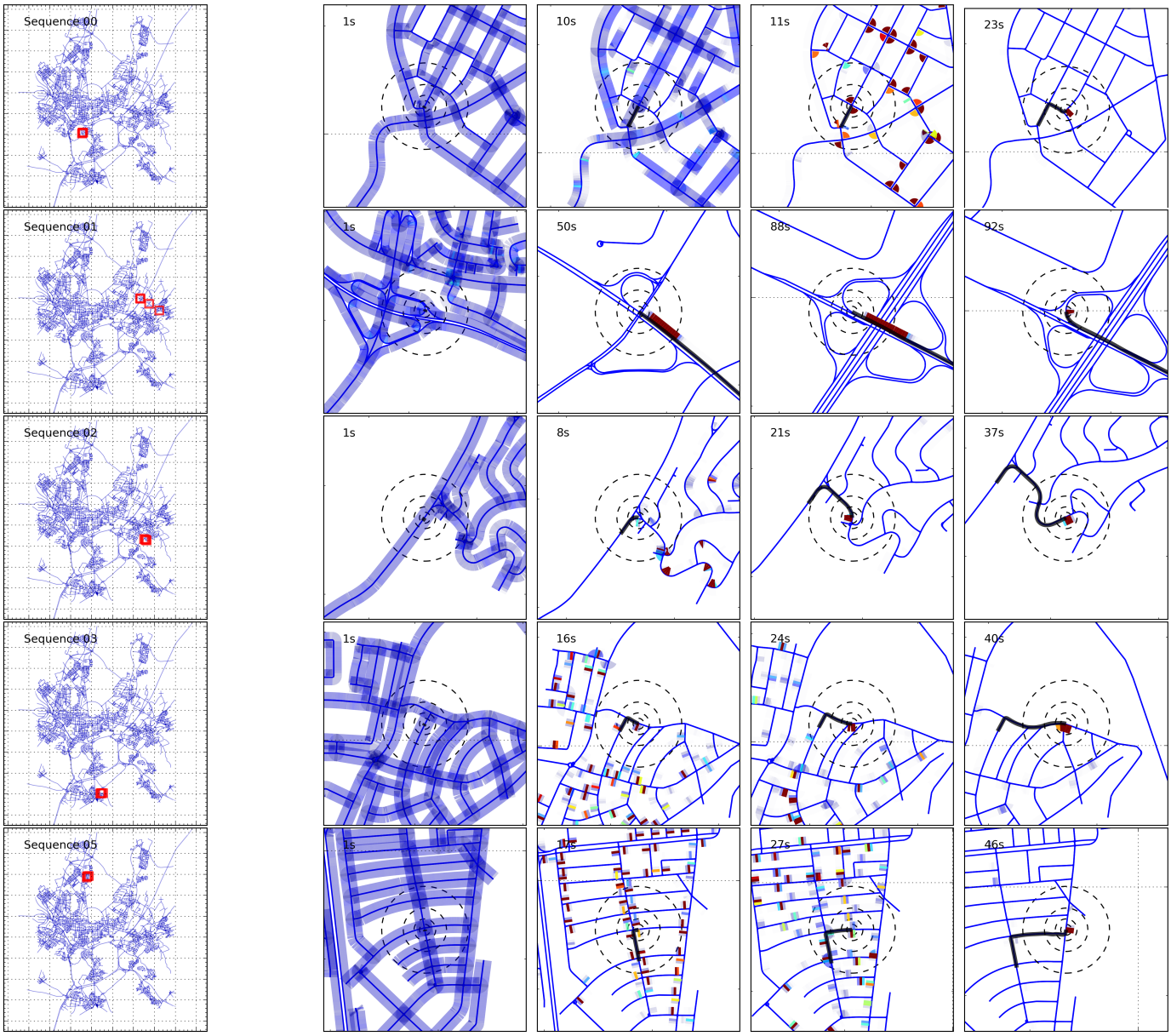

Fig. 11. **Selected Frames on the Full Map:** Inference results for unambiguous sequences. The left most column shows the full map for each sequence, followed by zoomed in sections of the map showing the posterior distribution over time. The black line is the GPS trajectory and the concentric circles indicate the current GPS position. Grid lines are every 2km. High probability is indicated with red while low probability regions are shown in blue.

and Mapping in the Space of Appearance," *IJRR*, vol. 27, no. 6, pp. 647–665, 2008.

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

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

[55] B.Williams, G. Klein, and I. Reid, "Automatic relocalization and loop closing for real-time monocular slam," *PAMI*, vol. 33(9), 2011.

[56] A. Javanmard, M. Haridasan, and L. Zhang, "Multi-track map matching," *ArXiv e-prints*, Sep 2012.

[57] D. Chen, A. Driemel, L. J. Guibas, A. Nguyen, and C. Wenk, "Approximate map matching with respect to the fréchet distance," in *ALENEX*, 2011, pp. 75–83.

[58] R. Raymond, T. Morimura, T. Osogami, and N. Hirosue, "Map matching with hidden markov model on sampled road network," in *ICPR*, 2012.

[59] Y. Li, Q. Huang, M. Kerber, L. Zhang, and L. Guibas, "Large-scale joint map matching of gps traces," in *SIGSPATIAL*, 2013.

[60] C. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on hidden markov model for real-time traffic sensing applications," in *ITSC*, 2012.

[61] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *GIS*, 2009.

[62] S. Se, D. Lowe, and J. Little, "Multi-track map matching," *Transactions on Robotics*, vol. 21(3), 2005.

[63] G. Floros, B. van der Zander, and B. Leibe, "OpenStreetSLAM: Global vehicle localization using OpenStreetMaps," in *ICRA*, 2013.

[64] E. Serradell, P. Glowacki, J. Kybic, F. Moreno-Noguer, and P. Fua, "Robust Non-Rigid Registration of 2D and 3D Graphs," in *CVPR*, 2012.

[65] B. Micusk, J. Koseck, and G. Singh, "Semantic parsing of street scenes from video," *IJRR*, vol. 31, no. 4, pp. 484–497, 2012.

[66] J. Hershey and P. Olsen, "Approximating the Kullback-Leibler Divergence Between Gaussian Mixture Models," in *ICASSP*, vol. 4, 2007.

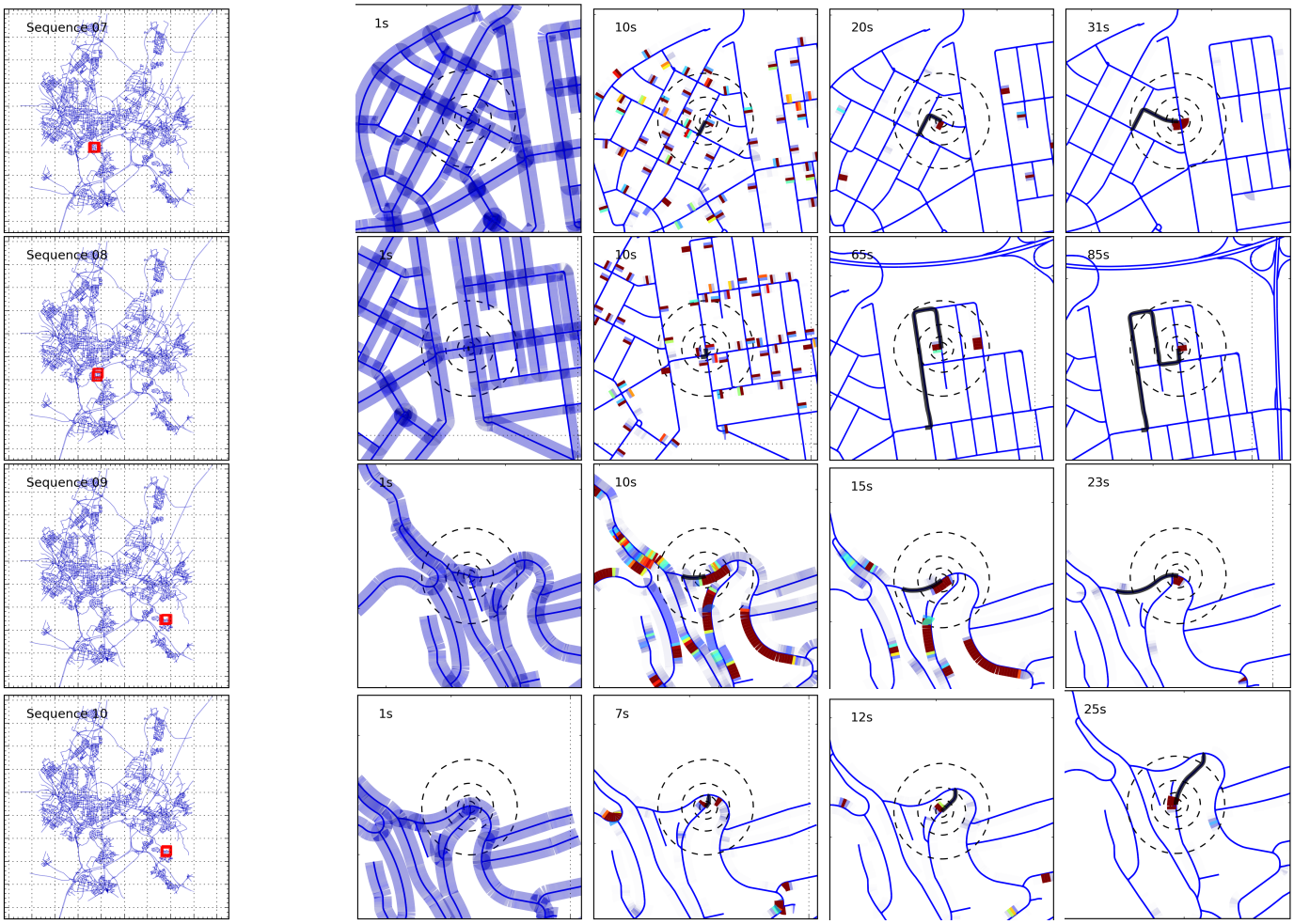

Fig. 12. **Selected Frames on the Full Map:** Inference results for unambiguous sequences. The left most column shows the full map for each sequence, followed by zoomed in sections of the map showing the posterior distribution over time. The black line is the GPS trajectory and the concentric circles indicate the current GPS position. Grid lines are every 2km. High probability is indicated with red while low probability regions are shown in blue.

Marcus A. Brubaker received his Ph.D. in Computer Science from the University of Toronto in 2011 and from 2011 to 2014 he was a postdoctoral fellow at TTI-Chicago. He is currently a postdoctoral fellow at the University of Toronto and consults with Cadre Research Labs. His research interests span statistics, machine learning and computer vision and includes applications in computational biology and forensics.

Andreas Geiger received his Diploma in computer science and his Ph.D. degree from Karlsruhe Institute of Technology in 2008 and 2013. Currently, he is a research scientist in the Perceiving Systems group at the Max Planck Institute for Intelligent Systems in Tübingen. His research interests include computer vision, machine learning and scene understanding.

Raquel Urtasun is an Assistant Professor at the Department of Computer Science, University of Toronto. From 2009-2014 she was an Assistant Professor at TTI-Chicago, a philanthropically endowed academic institute located in the campus of the University of Chicago. She was a visiting professor at ETH Zurich during the spring semester of 2010. Previously, she was a postdoctoral research scientist at UC Berkeley and ICSI and a postdoctoral associate at the Computer Science and Artificial Intelligence Laboratory (CSAIL) at MIT. Raquel Urtasun completed her PhD at the Computer Vision Laboratory, at EPFL, Switzerland in 2006 working with Pascal Fua and David Fleet at the University of Toronto. She has been area chair of multiple machine learning and vision conferences (i.e., NIPS, UAI, ICML, CVPR, ECCV, ICCV), she is on the editorial board of the International Journal of Computer Vision (IJCV), and served in the committee of numerous international conferences. Her major interests are statistical machine learning, computer vision and robotics, with a particular interest in structured prediction and their application to autonomous driving.