NOTES

HENRY O. JACOBS

Contents

1. Problem Description

Given a finite set of time-series of bounding boxes, representing the location of previously observed agents, and two successive bounding boxes of a newly observed agent, can we generate a time-dependent probability density, ρ, which represents the newly observed agent's location at times $t \in [0,T]$?

2. Our model

We will assume that agents are of two types, linear or nonlinear. Linear agents have dynamics specified by

$$\dot{x}(t) = x_0 + t \cdot v_0 \tag{1}$$

where $v_0, x_0 \in \mathbb{R}^2$ are the position and velocity at time $t = 0$. Nonlinear agents come in flavors $\{1,\dots,n\}$ have dynamics given by

$$\dot{x} = sX_k(x) \tag{2}$$

for some $k \in \{1,\dots,n\}$, a predetermined director-field, X_k, and some constant $s \in \mathbb{R}$. This motion model allows us to decompose the computation of $\rho(t)$ into

Date: December 31, 2016.

tractable components. In particular $\rho(t,x)$ is defined as the probability density of finding the agent at position x at time t given measurements of position and velocity $\hat{x}_0, \hat{v}_0$ at time $t = 0$. In terms of the density at time $t = 0$, ρ is given by

$$\rho(t,x) := \Pr(x = x(t) \mid \hat{x}_0, \hat{v}_0)$$

If we know the agent is linear and we know the velocity, then we know the position $x(t)$ by (1). If we know the agent is nonlinear, and we know the variables k and s, then we know the position $x(t)$ by integrating (2). These observations suggest that we decompose the computation of ρ using Baye's theorem, as We find

$$\begin{aligned}
\rho(t,x) &:= \int \Pr(x = x_t, x_0 \mid \mu) dx_0 \\
&= \sum_{k=1}^{n} \int \Pr(k, s, x = x_t, x_0 \mid \mu) ds\, dx_0 \\
&\quad + \int \Pr(x = x_t, \mathrm{Lin}, x_0, v_0 \mid \mu) dx_0 dv_0 \\
&= \Big(\sum_{k=1}^{n} \int \Pr(k, s, x = x_t, x_0, \mu) ds\, dx_0 \\
&\quad + \int \Pr(x = x_t, \mathrm{Lin}, x_0, v_0, \mu) dx_0 dv_0 \Big) / \Pr(\mu)
\end{aligned}$$

So we see we can compute ρ in terms of various joint probability distributions.

3. Efficient computation of the nonlinear component of ρ

Let us write ρ as

$$\rho(t,x) = \sum_k \int (\Phi_k^{st})_* \rho_{k,s}(x) ds + \mathit{linear\, term}$$

where $\rho_{k,s}(x) = \Pr(k, s, x \mid \hat{x}, \hat{v})$. If we only evaluate at discrete times, $t_n = n\Delta t$ for $\Delta t \ll 1$ then we need only compute

$$\rho(t_n,x) = \sum_k \int (\Phi_k^{sn\Delta t})_* \rho_{k,s}(x) ds + \mathit{linear\, term}.$$

We can approximate the integral over s with a Riemann sum with step-size $\Delta s_n = \bar{s}/n$, at integration nodes

$$\left\{ -\bar{s}, -\bar{s}\frac{n-1}{n}, \ldots, -\bar{s}\frac{1}{n}, 0, \bar{s}\frac{1}{n}, \ldots, \bar{s}\frac{n-1}{n}, \bar{s} \right\}$$

to yield the $\mathcal{O}(\Delta s_n)$ approximation

$$\rho(t_n, x) = \Delta s_n \left(\sum_k \sum_{|m| \leq n} (\Phi_k^{\bar{s} m \Delta t})_* \rho_{k,s}(x) \right) + \mathcal{O}(\Delta s_n) + linear\, term. \tag{3}$$

Finally, we may approximate $(\Phi_k^{st}) * \rho_{k,s}$ up to $\mathcal{O}(\Delta x)$ by using a grid Λ or resolution Δx. This gives us the approximation

$$(\Phi_k^{st})_* \rho_{k,s}(x) = |\Delta x| \sum_{\alpha \in \Lambda} \Pr(k, s, x_\alpha) \delta(x - \Phi_k^{st}(x_\alpha)) + \mathcal{O}(\Delta x)$$

Substitution of this approximation into (3) yields

$$\begin{aligned} &\rho(t_n, x) = \\ &\Delta s_n \cdot |\Delta x| \left(\sum_k \sum_{|m| \leq n} \sum_{\alpha \in \Lambda} \Pr(k, \bar{s} \frac{m}{n}, x_\alpha) \delta(x - \Phi_k^{\bar{s}\, m\, \Delta t}(x_\alpha)) \right) \\ &+ \mathcal{O}(\Delta s_n + \Delta x) + lin. \end{aligned} \tag{4}$$

From this computation it is clear what must be done. We need to compute the flow of the grid Λ at times $\bar{s} m \Delta t$ up to some maximal integer, n. Then we must compute $\Pr(k, \bar{s}\frac{m}{n}, x_\alpha)$ at each point on the grid and each m with $|m| \leq n$. Many of these computations of $\Pr(k, \bar{s}\frac{m}{n}, x_\alpha)$ could be stored, for example, when $n = 2^N$, about half of the computations overlap with the $n = 2^{N-1}$ case. This allows us to compute the above sum for all times $t_m \leq t_n$.

4. Choice of atomic distributions, and expressions for composite distributions

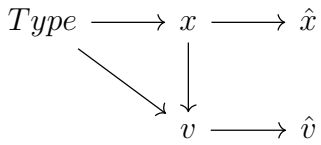
In order to compute $\Pr(k, s, x \mid \hat{x}, \hat{v})$ and $\Pr(x, v, \text{Lin} \mid \hat{x}, \hat{v})$ we use a probabilistic graphical model to organize our computations. To begin, define the set

$$Type = \{\text{Lin}\} \cup \{(k, s) \mid s \in [-\bar{s}, \bar{s}], k = 1, \ldots, n\}.$$

Each agent is associated with one and only one type. The type of a given agents provides information about his true position x based on how often we've see an agent of such a type at location x. Moreover, given an agents true position, and his type, we know something about his/her velocity. For example, if an agent is of type (k, s) and is known to be at position x, then we know with certainty that his velocity is $v = sX_k(x)$. Lastly, we assume our measurements of position are independent of Type, given the true position. This yields the probabilistic

graphical model

(5)



Under this model, the posteriors $\Pr(k, s, x \mid \hat{x}, \hat{v})$ and $\Pr(x, v, \mathrm{Lin} \mid \hat{x}, \hat{v})$ can be computed in terms of the *atomic distributions*

- $\Pr(x \mid \mathrm{Lin}) \sim \mathcal{U}([-w/2, w/2] \times [-h/2, h/2])$
- $\Pr(v \mid \mathrm{Lin}, x) \sim \mathcal{N}(0, \sigma_{\mathrm{Lin}})$
- $\Pr(\hat{x} \mid x) \sim \mathcal{N}(x, \sigma_x)$
- $\Pr(\hat{v} \mid v) \sim \mathcal{N}(x, \sigma_v)$
- $\Pr(Type)$

These atomic distributions are chosen, and represent a set of parameters for our model. In the remainder of this section, we will assume that these atomic densities are known, and easily computable, expressions. We list precise choices for the atomic densities in the appendix.

To calculate $\Pr(k, s, x \mid \hat{x}, \hat{v})$ in terms of the atomic distribution we use (5) to find

$$\begin{aligned}\Pr(k, s, x \mid \hat{x}, \hat{v}) &= \frac{\Pr(k, s, x, \hat{x}, \hat{v})}{\Pr(\hat{x}, \hat{v})} \\ &= \frac{\Pr(k, s, x, \hat{x}, \hat{v})}{\Pr(\hat{x}, \hat{v})} \\ &= \frac{\int \Pr(k, s, x, v, \hat{x}, \hat{v}) dv}{\Pr(\hat{x}, \hat{v})}\end{aligned}$$

Using the pgm diagram we can decompose the integrand in the numerator as

$$\begin{aligned}&= \frac{1}{\Pr(\hat{x}, \hat{v})} \int \Pr(x \mid k, s) \Pr(\hat{x} \mid x) \Pr(\hat{v} \mid v) \Pr(v \mid k, s, x) \Pr(k, s) dv \\ &= \frac{\Pr(x \mid k, s) \Pr(\hat{x} \mid x) \Pr(\hat{v} \mid v = sX_k(x)) \Pr(k) \Pr(s)}{\Pr(\hat{x}, \hat{v})}\end{aligned}$$

In order to compute $\Pr(\hat{x}, \hat{v})$ we can compute

$$\Pr(\mathrm{Lin}, x \mid \hat{x}, \hat{v}) = \frac{\Pr(\mathrm{Lin}, x, \hat{x}, \hat{v})}{\Pr(\hat{x}, \hat{v})}$$

where

$$\Pr(\text{Lin}, x, \hat{x}, \hat{v}) = \int \Pr(x \mid \text{Lin}) \Pr(\hat{x} \mid x) \Pr(\hat{v} \mid v) \Pr(v \mid \text{Lin}, x) \Pr(\text{Lin}) dv$$
$$= \chi_{D_x}(x) G_{\sigma_x}(x - \hat{x}) \Pr(\text{Lin}) \int G_{\sigma_v}(v - \hat{v}) G_{\sigma_{\text{Lin}}}(v) dv$$

Then we may use the fact that

$$\Pr(\hat{x}, \hat{v}) = \int \Pr(\text{Lin}, x, \hat{x}, \hat{v}) dx + \sum_k \int \Pr(k, s, x, \hat{x}, \hat{v}) dx ds$$

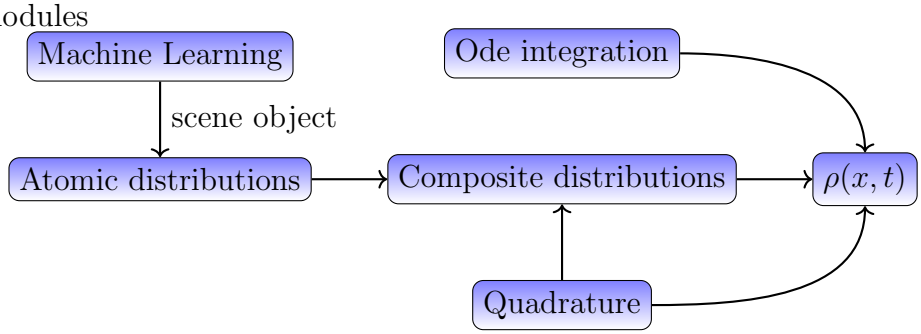
Finally, we should note that

$$\Pr(\text{Lin}, x, v \mid \hat{x}, \hat{v}) = \frac{\Pr(\text{Lin}, x, v, \hat{x}, \hat{v})}{\Pr(\hat{x}, \hat{v})}$$

where the numerator is

$$\Pr(\text{Lin}, x, v, \hat{x}, \hat{v}) = \Pr(x \mid \text{Lin}) \Pr(\hat{x} \mid x) \Pr(\hat{v} \mid v) \Pr(v \mid \text{Lin}, x) \Pr(\text{Lin}).$$

4.1. **Code design.** The ultimate goal is to compute $\rho(x, t)$, or at least to integrate over a region of space-time. To do this we can consider the following diagram of modules



Each arrow in the above diagram corresponds to an import or a load command. The names of each module should convey what sort of computations the module is responsible for. For example, the Machine learning module processes the data from the Stanford drone data set, and outputs a scene object. This scene object includes variables $\theta, \alpha, P(k), P(s), P(\text{Lin})$ and the max bounding box width. where θ is a 3d array which allows one to compute the director field for each class and α is a 3d array which allows one to compute the potential function involved in the atomic distribution $\Pr(x_0 \mid k)$. The usage of θ and α is described in appendix B.

5. Evaluation

We would like to view the computed ρ as a classifier. Let $\tau > 0$. Given a partition of measurable sets, $\mathcal{E}$, we consider the classifier $pred : \mathcal{E} \to \{+, -\}$ given

by

$$pred(E) = \int_E \rho > \tau$$

If we have a ground truth distribution ρ_{true} we define the variable

$$truth(E) = \int_E \rho_{true} > \tau.$$

If we weight each of the sets of this partition we can define precision, accuracy, and recall as

$$\begin{aligned} precision &= \Pr_{\mathcal{E}}(pred \mid truth) \\ recall &= \Pr_{\mathcal{E}}(truth \mid pred) \\ accuracy &= \Pr_{\mathcal{E}}(truth = pred) \end{aligned}$$

We can use these as evaluation metrics. Unfortunately, that we get different metrics for different τs means we must do this for a variety of them.

Appendix A. Atomic distributions

Let σ_x be an estimate of the variance of our position measurement. It follows that if velocity is measured simply as the difference between two consecutive position measurements, then the variance of our velocity measurement is $2\sigma_x$. Similarly, let $V_k : \mathbb{R}^2 \to \mathbb{R}$ be an energy function and $X_k \in \mathfrak{X}(\mathbb{R}^2)$ a director field we learn from the data. (see Appendix B) The atomic distributions, $\Pr(k)$, $\Pr(s)$, and $\Pr(\text{Lin})$ are actually computed in the Machine Learning module and then imported. Here are posteriors:

$$\begin{aligned} \Pr(\hat{x}_0 \mid x_0) &\sim \mathcal{N}(x_0; \sigma_x) \\ \Pr(\hat{v}_0 \mid v_0) &\sim \mathcal{N}(v_0; 2\sigma_x) \\ \Pr(x_0 \mid k, s) = \Pr(x_0 \mid k) &= \frac{1}{Z_k} \exp\left(-V_k(x_0)\right) \\ \Pr(v_0 \mid k, s, x_0) &= \delta(v_0 - sX_k(x_0)) \end{aligned}$$

The only remaining posterior to provide is $\Pr(x \mid \text{Lin})$ and $\Pr(v \mid x, \text{Lin})$. We let $\Pr(x_0 \mid \text{Lin})$ be a uniform distribution over our domain and we let

$$\Pr(v_0 \mid x_0, \text{Lin}) = \Pr(v_0 \mid \text{Lin}) \sim \mathcal{N}(0, \sigma_{\text{Lin}}) \tag{6}$$

where $\sigma_{\text{Lin}} > 0$ is learned from the data.

Appendix B. The Machine Learning Module

The machine learning module is responsible for clustering the trajectories, and providing the vector-fields and potential functions for each cluster, as well as the probability $\Pr(k), \Pr(s)$ and $\Pr(\text{Lin})$ and the parameter σ_{Lin}.

Appendix C. Posteriors

Given our choices for the atomic expressions, we can derive expressions for some of the relevant posterior expressions. In particular,

$$\begin{aligned}
\Pr(\vec{x}_t, \text{Lin} \mid \mu) &= \int \Pr(\vec{x}_t, \vec{v}_0, \text{Lin} \mid \mu) dv_0 \\
&= \frac{1}{\Pr(\mu)} \int \Pr(\vec{x}_t, \vec{x}_0, \vec{v}_0, \text{Lin}, \mu) d\vec{v}_0 \, d\vec{x}_0 \\
&= \frac{\Pr(\text{Lin})}{\Pr(\mu)} \int \Pr(\vec{x}_t \mid \vec{x}_0, \vec{v}_0, \text{Lin}) \Pr(\hat{x}_0 \mid x_0) \\
&\qquad\qquad \Pr(\hat{v}_0 \mid v_0) \Pr(v_0 \mid \text{Lin}) \Pr(x_0 \mid \text{Lin}) d\vec{v}_0 \, d\vec{x}_0 \\
&= \frac{\Pr(\text{Lin})}{\Pr(\mu)} \int \chi(\vec{x}_t - t\vec{v}_0 \in D) \Pr(\hat{\vec{x}}_0 \mid \vec{x}_0 = \vec{x}_t - t\vec{v}_0) \Pr(\hat{\vec{v}}_0 \mid \vec{v}_0) \Pr(\vec{v}_0 \mid \text{Lin}) d\vec{v}_0 \\
&= A \cdot I_1 \cdot I_2
\end{aligned}$$

where

$$\begin{aligned}
A &= \frac{\Pr(\text{Lin})}{\Pr(\mu)} \cdot (w \cdot h \cdot 8\pi^3 \sigma_{\text{Lin}}^2 \sigma_x^2 \sigma_v^2)^{-1} \\
I_1 &= \int_{(x_t - w/2)/t}^{(x_t + w/2)/t} \exp\left(\frac{-(\hat{x}_0 - x_t - tu)^2}{2\sigma_x^2} - \frac{(\hat{u}_0 - u)^2}{8\sigma_x^2} - \frac{u^2}{2\sigma_{\text{Lin}}^2} \right) du \\
I_2 &= \int_{(y_t - h/2)/t}^{(y_t + h/2)/t} \exp\left(\frac{-(\hat{y}_0 - y_t - tv)^2}{2\sigma_x^2} - \frac{(\hat{v}_0 - v)^2}{8\sigma_x^2} - \frac{v^2}{2\sigma_{\text{Lin}}^2} \right) dv
\end{aligned}$$

We can express the integral terms, I_1 and I_2, in terms of the error function. In particular, the logarithm of the integrand is just a quadratic function of the integration variable. We can complete the square and write the exponent as

$$-a(u - \bar{u})^2 + k$$

where

$$\begin{aligned}
a &= \frac{4t^2 + 1}{2\sigma_x^2} + \frac{1}{2\sigma_L^2} \\
\bar{u} &= \frac{\hat{u} + 4\hat{x}t - 4tx_t}{4t^2 + 1 + 4(\sigma_x/\sigma_L)^2} \\
k &= \frac{1}{\sigma_x^2 \left(4\sigma_L^2 t^2 + \sigma_L^2 + 4\sigma_x^2\right)} \cdot \\
&\quad \Big(-\frac{\hat{u}^2 \sigma_L^2}{2} t^2 - \frac{\hat{u}^2 \sigma_x^2}{2} + \hat{u}\hat{x}\sigma_L^2 t \\
&\qquad - \hat{u}\sigma_L^2 t x_t - \frac{\hat{x}^2 \sigma_L^2}{2} - 2\hat{x}^2 \sigma_x^2 + \hat{x}\sigma_L^2 x_t + 4\hat{x}\sigma_x^2 x_t - \frac{\sigma_L^2 x_t^2}{2} - 2\sigma_x^2 x_t^2 \Big)
\end{aligned}$$

So that

$$I_1 = \left[\exp(k) \frac{\sqrt{\pi}}{2\sqrt{a}} \operatorname{erf}(\sqrt{a}(u - \bar{u})) \right] \Big|_{u=(x_t - w/2)/t}^{(x_t + w/2)/t}$$

A similar expression holds for I_2 by substituting $\hat{x}$ with $\hat{y}$ and w with h, etc.

C.0.1. *Learning clusters.* For a fixed scene with a database of agent trajectories we cluster the trajectories by applying the Affinity propagation algorithm to the end-points. We then prune the clusters by discarding trajectories which are outliers with respect to total length (we define an outlier using the standard inter-quartile range criterion with a IQR coefficient of 1.5). We then throw out clusters which contain less than 10% of the trajectories. We associate class labels, $1, \ldots, n$, to the remaining clusters, and we will develop a model for each of these classes in the next section. We also add an additional class, "Lin", where the underlying model will be a linear predictor. Finally, we define a prior, $P(c)$ to compute the probability that a given agent falls within one of these classes. We set

$$P(\text{Lin}) = \frac{\#\text{ discarded trajectories}}{\#\text{ trajectories}}$$
$$P(k) = \frac{\#\text{ trajectories in cluster } k}{\#\text{ trajectories}} \text{ for } k = 1, \ldots, n.$$

C.0.2. *Learning $P(x \mid k)$.* During runtime we will need this computation to be fast, and to scale with the size of the data-set. This rules out standard probabilistic classification schemes such kernel density methods. Instead we discretize the space of energy functions, and compute on a subspace of fixed dimension. For the linear-predictor class, Lin, we assume $P(x \mid \text{Lin})$ is a uniform distribution over our domain. For $k = 1, \ldots, n$ we use the data to learn $P(x \mid k)$. Given data points $x_{1,k}, \ldots, x_{N,k}$ associated to class k we define

$$P(x \mid k) := \frac{1}{Z_k} e^{-V_k(x)} \tag{7}$$

where

$$V_k = \operatorname{argmin}_{V \in H_n} \left(\log \left(\int e^{-V(x)} dx \right) + \sum_{i=1}^{N} V(x_{i,k}) \right).$$

over some finite-dimensional subspace, $H_n \subset C^0(\mathbb{R}^2)$ (perhaps a space of low order polynomials). This is a justifiable choice, since $V_k(x)$ is the most likely potential function in H_n, if the observations $\{x_{i,k}\}_{k=1}^N$ are drawn from (7). Again, the advantage to this approach is that we may restrict V_k to a class of quickly computable functions which will not slow down performance at runtime. In my current implementation, H_n is a sum of tensor products of low order Legendre polynomials.

C.0.3. *Learning vector fields.* Given trajectories, we may fit a director field of the form $X(x,y) = (\cos(\theta_\alpha(x,y)), \sin(\theta_\alpha(x,y)))$ where

$$\theta_\alpha(x,y) = \sum_{ij} \alpha_{ij} L_i(x/w) L_j(y/h).$$

Given observations $\vec{x}_0, \ldots, \vec{x}_n \in \mathbb{R}^2$ we may compute a series of unit vectors, $\vec{u}_k = \Delta\vec{x}_k / \|\Delta\vec{x}_k\|$, where $\Delta\vec{x}_k = \vec{x}_{k+1} - \vec{x}_k$. A director-field may be learned from these directions by maximizing

$$R(\alpha) = \sum_k \frac{x_{k+1} + x_k}{2} \cos(\theta_\alpha(\vec{x}_k)) + \frac{y_{k+1} + y_k}{2} \sin(\theta_\alpha(\vec{x}_k)).$$

In words, this reward function measures the alignment of the director-field with an observation (i.e. the dot product), and takes sum over all observations.

Appendix D. Computation of $\rho(x,t)$

Assuming we have reliable quadrature and ode integration routines, we should be able to compute $\rho(x,t)$ efficiently. In particular,

$$\rho(x,t) = \rho_{\text{Lin}}(x,t) + \sum_k \rho_k(x,t)$$

where

$$\begin{aligned}\rho_k(x,t) &= \int_{-s_{\max}}^{s_{\max}} \Pr(x_0 = (\Phi_k^{st})^{-1}(x), k, s \mid \mu) \det(D\Phi_k^{st}(x)^{-1}) ds \\ &= \int_{-s_{\max}}^{s_{\max}} (\Phi_k^{st})_* \rho_{k,s}(x) ds.\end{aligned}$$

where $\rho_{k,s}(x) = \Pr(x_0 = x, k, s \mid \mu)$. For $t = j \cdot \Delta t$ we can use a Riemann sum to approximate the above quadrature as

$$\rho_k(x, j\Delta t) = \sum_{\ell=-j}^{j} (\Phi_k^{t_\ell})_* \rho_{k,s_\ell}(x) \Delta s + \mathcal{O}(\Delta s).$$

where

$$\begin{aligned}t_\ell &= s_{\max}\, \ell\, \Delta t \\ \Delta s &= s_{\max}\, \Delta t \\ s_\ell &= \ell\, s_{\max}/j.\end{aligned}$$

Of course, we can use higher order quadrature rule if we desire greater orders of accuracy in s. The advantage of the above formula is that we may recycle our computation of the flow, $\Phi_k^{t_\ell}$, as j increases. The push-forward distributions cab

be approximated using Riemann quadrature, as sums of Dirac delta distributions. In particular

$$\Phi_* \rho_{k,s} = \|\Delta \vec{x}\| \cdot \sum_{x \in \Gamma} \rho_{k,s}(x) \delta_{\Phi(x)} + \mathcal{O}(\|\Delta \vec{x}\|).$$

where Γ is a regular grid, overlaid on the support of $\rho_{k,s}$ with a volume element $\|\Delta \vec{x}\|$.

This gives us the following pipeline for plotting $\rho_k(x,t)$ for $t \in 0, \Delta t, \ldots, j \cdot \Delta t$.

Data: scene, $\mu, N_t, \Delta t$
Result: $\rho_k(t)$ for $t = 0, \Delta t, \ldots, N_t \, \Delta t$
Initialize Γ_0 and $V_0 = 1$;
for $j = 1, \ldots, N_t$ **do**
  Compute $\Gamma_j^+, V_j^+ = F_k(\Gamma_{j-1}^+, V_{j-1}^+; \Delta t)$;
  Compute $\Gamma_j^-, V_j^- = F_k(\Gamma_{j-1}^-, V_{j-1}^-; -\Delta t)$;
  Let $\rho_k = \sum_{x \in \Gamma_0} \Pr(x_0 = x_\alpha, k, s = 0 \mid \mu) \delta_x$;
  for $\ell = 1, \ldots, j$ **do**
    Compute $p_\alpha^+ = \Pr(x_0 = x_\alpha, k, s_\ell \mid \mu) \Delta s$ for $x_\alpha \in \Gamma_0$;
    $\rho_k + = \sum_\alpha p_\alpha^+ \delta_{x_{\alpha,\ell}}$ for $x_{\alpha,\ell} \in \Gamma_\ell^+$;
    Compute $p_\alpha^- = \Pr(x_0 = x_\alpha, k, -s_\ell \mid \mu) \Delta s$ for $x_\alpha \in \Gamma_0$;
    $\rho_k + = \sum_\alpha p_\alpha^- \delta_{x_{\alpha,\ell}}$ for $x_{\alpha,\ell} \in \Gamma_\ell^-$;
  end
  return ρ_k
end

Appendix E. Numerical analysis for our particle method

Let $\rho_0 \in W^{1,1}(\mathbb{R}^n)$ with a support contained in a compact rectangular domain D which we uniform ally grid. By the Sobelev embedding theorem we know that $\rho_0 \in L^\infty$. Let $\Gamma \subset \mathbb{R}^n$ be a regular grid on D with resolution $\Delta \vec{x}$ and volume element $\|\Delta \vec{x}\|$. Then

$$\left\| \rho_0 - \|\Delta \vec{x}\| \cdot \sum_{x \in \Gamma} \rho_0(x) \delta_x \right\|_{L^1} = \mathcal{O}(\|\Delta \vec{x}\|).$$

Therefore

$$\Phi_* \rho_0 = \|\Delta \vec{x}\| \cdot \sum_{x \in \Gamma} \rho_0(x) \delta_{\Phi(x)} + \mathcal{O}(\|\Delta \vec{x}\|).$$

for any C^1 diffeomorphism Φ. *Note: when you plot this density on your computer screen, it is tempting to just plot a bunch of dots at $\Phi(x)$ for $x \in \Gamma$, weighted by $\rho_0(x)$, then linearly interpolate. However, this is not what the eye expects (although it has the correct information). When people "draw a picture of a distribution", μ, they are not really drawing a distribution (i.e. an element of a dual space to C^0). They are typically drawing a Radon-Nikodym derivative with respect to the*

Lebesgue measure. In this case, before plotting, we should divide the values of each point by the amount of expansion witnessed at that point, $\det(D\Phi(x))$*. It's a subtle point, since there is a lot of human psychology going on here. The plotted object is a manipulated version of the object under scrutiny.*