1 Introduction to wavelet density estimation

Wavelet density estimation was a promising technique for shape matching, as explored by Peter and Rangarajan in their 2008 paper[**?**]. Generally, the goal is to estimate a probability density function over the shape points and use that representation to compare shapes. The density is estimated by expansion into a wavelet basis form, which has a theory with strong mathematical foundations and provides several advantages over competing function bases (e.g. the Fourier basis, or simple histograms).

This wavelet density estimation framework can be applied for any dimensional data (e.g. 1D, 2D, and 3D shapes) by using the tensor product (multidimensional) forms of the 1D wavelet bases. The coefficients of this functional basis (of the probability density function) form our feature representation, and also conveniently map to the unit hypersphere due to constraints we impose during density estimation. This manifold geometry allows us to use a simple, efficient, and well-defined distance metric as our shape dissimilarity measure.

2 Implementation and our optimization

The wavelet density estimator code by Peter and Rangarajan accurately constructs the coefficient vectors and thus estimates the densities, but because of the sheer complexity of the procedure, there was significant room to improve the runtime. A full estimation over the standard shape databases can take multiple days even with relatively low resolution. For instance, the Brown database has 99 shapes, the Swedish Leaf database has 1125 shapes, and the MPEG7 database has 1400 shapes, which are quite large already; the Princeton ModelNet database is orders of magnitude larger, at 127,915 shapes. At these scales, the inefficiency of the Peter-Rangarajan code is obvious (i.e. ModelNet estimation becomes intractable). Optimization is thus a primary priority.

THe Peter-Rangarajan follows the full mathematical implementation, which means there exist superfluous calculations. For instance, they loop over every sample point and evaluate every basis function individually. We added an optimization of only calculating the basis functions for relevant sample points under specified basis functions with the hope to cut computation power (and thus, time required for estimation) considerably.

In this report we first provide an exposition of wavelets and its use in density estimation, delving deeper into the equations which drive the core of the implementation by Peter and Rangarajan. We then provide an explanation of the optimizations, and finally report the extremely promising results from said optimizations.

Wavelet Density Estimation

Our feature representation is formed by estimation of the probability density function of the shape, which we model as sample points from a distribution.

First, we must explain the use of wavelets as the foundation of our density estimation framework. An L^2 function, also known as a square-integrable function, is a real- or complex-valued function whose square integrates to a finite number. Probability density functions fall under this function space, therefore allowing us to use wavelets for density estimation using a linear combination of these basis functions. In other words, wavelet density estimation as a paradigm is built on strong mathematical foundations.

The equation for 1D density estimation is

$$p(x) = \sum_{j_o,k} \alpha_{j_o,k}\phi_{j_o,k}(x) + \sum_{j \geq j_o,k}^{j_1} \beta_{j,k}\psi_{j,k}(x) \tag{1}$$

where $\phi(x)$ and $\psi(x)$ are the scaling and wavelet basis functions, also known as the "father" and "mother" wavelets, respectively. The functions can also be set to some starting resolution level that determines its dilation and contraction. To extend to multiple resolutions, we incorporate the wavelet basis function with a starting (j_o) and stopping (j) resolution level. In order to form a basis, we need to make multiple copies of the function over different translates, k. These functions are scaled by their scaling and wavelet basis function coefficients, $\alpha_{j_o,k}$ and $\beta_{j,k}$. To easily retain properties of true densities, such as non-negativity and integration to one, we compute the square root of the probability density function

$$\sqrt{p(x)} = \sum_{j_o,k} \alpha_{j_o,k}\phi_{j_o,k}(x) + \sum_{j \geq j_o,k}^{j_1} \beta_{j,k}\psi_{j,k}(x) \tag{2}$$

Since we optimized for multi-resolution 2D wavelet density estimation, we consider the second dimension with $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ and $\mathbf{k} = (k_1, k_2) \in \mathbb{Z}^2$. We also extend the equation to

$$\sqrt{p(\mathbf{x})} = \sum_{j_o,\mathbf{k}} \alpha_{j_o,\mathbf{k}}\phi_{j_o,\mathbf{k}}(\mathbf{x}) + \sum_{j \geq j_o,\mathbf{k}}^{j_1} \sum_{w=1}^{3} \beta_{j,\mathbf{k}}^{w}\psi_{j,\mathbf{k}}^{w}(\mathbf{x}) \tag{3}$$

where we calculate for the basis functions by using the tensor product method

$$\phi_{j_o,\mathbf{k}}(\mathbf{x}) = 2^{j_o}\phi(2^{j_o}x_1 - k_1)\phi(2^{j_o}x_2 - k_2)$$
$$\psi^1_{j,\mathbf{k}}(\mathbf{x}) = 2^{j}\phi(2^{j}x_1 - k_1)\psi(2^{j}x_2 - k_2)$$
$$\psi^2_{j,\mathbf{k}}(\mathbf{x}) = 2^{j}\psi(2^{j}x_1 - k_1)\phi(2^{j}x_2 - k_2)$$
$$\psi^3_{j,\mathbf{k}}(\mathbf{x}) = 2^{j}\psi(2^{j}x_1 - k_1)\psi(2^{j}x_2 - k_2) \tag{4}$$

A mathematical advantage in using wavelets is its compact support, meaning they are nonzero on a small domain. Any point falling outside of the basis function's support does not contribute to the basis function value at that translation.

Intuitively, the scaling and wavelet basis function coefficients determine the height of its basis function. These coefficients thus uniquely define the probability density function of a shape, which means we can use them as our feature representation in shape retrieval. In order to calculate coefficients we use the following equations:

$$\alpha_{j_o,\mathbf{k}} = \left(\frac{1}{N}\right)\sum_{i=1}^{N}\frac{\phi_{j_o,\mathbf{k}}(\mathbf{x})}{\sqrt{p(\mathbf{x})}} \tag{5}$$

$$\beta_{j_o,k} = \left(\frac{1}{N}\right)\sum_{i=1}^{N}\frac{\psi_{j,\mathbf{k}(\mathbf{x})}}{\sqrt{p(\mathbf{x})}} \tag{6}$$

As we estimate the coefficients of our probability density function, we use a loss function to determine whether our coefficients are as close to optimal as possible. To do this we try to minimize a negative loglikelihood objective function with respect to the coefficients

$$-\log p(X;\{\alpha_{j_o,k}\beta_{j,k}\}) = -\frac{1}{N}\sum_{i=1}^{N}\log\left[\sum_{j_o,k}\alpha_{j_o,k}\phi_{j_o,k}(x_i) + \sum_{j\geq j_o,k}^{j_1}\beta_{j,k}\psi_{j.k}(x_i)\right]^2 \tag{7}$$

where $X = \{x_i\}_{i=1}^N$. To determine the direction of the gradient descent, we compute the following expression:

$$-2\left(\frac{1}{N}\right)\sum\frac{\phi_{j_o,\mathbf{k}}(\mathbf{x})}{\sum\alpha_{j_o,\mathbf{k}}\phi_{j_o,\mathbf{k}}(\mathbf{x})} \tag{8}$$

In order for maintain essential properties of density estimation, we constrain the coefficients to one

$$\sum_{j_o,k}\alpha^2_{j_o,k} + \sum_{j\geq j_o,k}^{j_1}\beta^2_{j,k} = 1 \tag{9}$$

This constraint conveniently maps our feature representation into a unit hypersphere which allows us to take advantage of its geometric properties to signify similarity and dissimilarity between shapes in shape retrieval.

Wavelet Density Estimation Optimization

Now that we have an overview of wavelet density estimation and the equations used, we can delve into our optimization. We focus our efforts on the 2D form of the density estimation on a given point set shape representation. We then use the coefficients that uniquely characterize the density function as our feature vector.

Originally, for a database such as MPEG7, which has 1400 shapes, an estimation process with a domain of [-0.5,0.5], resolution level 2 to 3, and 1344 translates would take roughly 16.8 hours. This is a particularly low resolution, and if raised we would expect an even slower runtime. To optimize this code for speed, we focus on the bottleneck: the calculation of the initial coefficients and negative log likelihood cost value with its gradient.

Initializing Coefficients

This function calculates equations (5) and (6) to compute initial coefficients that will be updated later through a loss-minimizing optimization process. The time spent for this calculation with the same parameters mentioned above is roughly around 1.8 of the 16.8 hours.

Given a point set shape representation, the wavelet density estimation framework covers the shape with basis functions at each translation. Each function has a coefficient that needs to be calculated. The bottleneck exists because of the way the code structures the computation: it code computes equations (4) for a *single point over all*

translations using a a Kronecker tensor product. This means that if we have 1344 translations, it would perform 1344 operations for a single point. If a shape is made up of 4007 sample points, then it would be performing a total of 5,385,408 operations for a single shape. This calls for an excess amount of redundant computations since most of the scaling and wavelet basis functions for a single point over all translations return the value of zero because the observed sample point does not contribute to most basis functions, i.e. does not exist under the compact support of the basis function at specific translations. The solution is to change the structure of the looping:

Instead of looping through each point and finding which basis functions it falls under, we looped through basis functions along the horizontal direction, evaluated its basis function value (4) based on the points fall under its support, and placed them into a matrix that holds these values. We then store this matrix to be used for later optimized computation (for the negative log likelihood cost value and gradient). Once we have this matrix of values, we can evaluate the basis function coefficients with the relevant points under each translation using equation (5) and (6).

Negative Log Likelihood

The second area where the bottleneck occurs is in computing for the negative log likelihood cost value (7) and the function's gradient (8). We use these equations to check whether we have found the optimal coefficient values and which direction in which to move to optimize. The original time spent for this computation was around 13.9 of the 16.8 hours.

The original code actually replicated much of the work done in initialize coefficients. It would take a single point and calculate the basis function over all translations, resulting in needless computation. It does this to attain a matrix of the basis function values to perform the appropriate operations to solve for the cost value and gradient.

Since we already performed most of the heavy computation to attain this matrix in initializing coefficients, our solution was to simply pass in the needed matrix of basis function values for each translation and perform the proper computations. This effectively solves for the negative log likelihood cost value and gradient while ultimately eliminating loops.

Results

As mentioned above, our goal is to estimate a density function of a 2D shape using wavelets. This allows us to extract the coefficients as our feature representation to be used in shape retrieval on a unit hypersphere. However, with MPEG7 containing 1400 shapes, domain [-0.5,0.5], resolution level 2 to 3, and 1344 translates, it would take about 16.8 hours to calculate the coefficients. Our optimization yields significant improvements over the original code.

The original time it took the code to calculate for only the initial coefficients was about 1.8 hours. After our optimization, the run time to compute the initial coefficients cut all the way down to 0.13 hours, or about eight minutes. Ultimately, the computation runs 92.8% faster. As for calculations for the negative log likelihood cost value and gradient, our optimization shaved the lines of code from 54 lines to 21 lines, ridding it of loops. We essentially do away with 61% of lines of code. The run time was promising as well. To calculate the cost value and gradient for seven iterations, computation time went from 13.9 hours to 0.089 hours, or around 5 minutes for the entire dataset. The optimized computation runs 99% faster. Overall, to estimate the wavelet densities of a database of 2D shapes with the optimized code under the specified parameters would take 0.3 hours, or 17.8 minutes, running 98% faster.

With this optimization, we hope that researchers using the wavelet density framework can achieve full estimation of the coefficient values quickly, and spend more time in developing new and better methods of shape retrieval.