1 Implementation and our optimization

The wavelet density estimator code by Peter and Rangarajan accurately constructs the coefficient vectors (the alphas and betas as seen in **??**) 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 time required for estimation.

2 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.

The resolution levels and domain size controls the number of translations. Originally, for a database such as MPEG7, which has 1400 shapes, an estimation process with a domain of [-0.5,0.5] in the both x and y directions, resolution levels 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.

2.1 Initializing Coefficients

This function calculates equations (**??**) and (**??**) 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 (**??**) for a *single point over all translations* using a Kronecker tensor product [**?**]. This means that if we have 1344 translations, it would perform 1344 operations for a single sample 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 (**??**) based on the points that 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 (**??**) and (**??**).

2.2 Negative log likelihood

The second area where the bottleneck occurs is in computing for the negative log likelihood cost value (**??**) and the function's gradient (**??**). 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.

3 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 about 100 times 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 per shape, computation time went from 13.9 hours to 0.089 hours, or around 5 minutes for the entire dataset. The optimized computation runs 100 times 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 a little under 100 times 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.