On-line Learning of Parametric Mixture Models for 2D Simulation

Yuan Tian
Department of Computer Science
Dartmouth College

***Abstract*—Monte Carlo techniques are used to construct light transport paths. One way to improve the light transport simulation in the complex scenes is to recover the sampling distributions from particles distributed in the scene prior to rendering process. Some researchers have proposed to represent the distributions by a parametric mixture model, which has a good performance when use these distributions to recover sampling distributions in the complex lighting scenes. To recover this procedure and find the more accurate regression model, we use the 2D image as the training example and tried the different schemes to improve the regression model. Using the resulted regression model we can simulate the 2D image more accurate.**

I. Introduction

Efficient light transport simulation is a challenge problem, especially for the problems in highly occluded scenes, where the probability of obtaining a non-zero contribution light path is very low. To solve this problem, researchers think about getting the local scattering directions and light emissions samples firstly, and then use it when constructing light transport paths. In this way, the probability of constructing light paths with non-zero contributions is increased and it also reduces the variance without bias introduced.

Because the inflexible representations of distributions [1] and the methods rely on a limited number of particles are usually insufficient to recover useful sampling distributions in highly occluded scenes. The researchers proposed to represent the sampling distributions with Gaussian mixture model(GMM) [2], extensively used in machine learning [3]. Because the GMM has many beneficial properties in learning, evaluating, sampling and also the storage. Their approach consists of two steps: training and rendering. During the training step they shoot particles from lights and the sensor in a scene and model importance and radiance density functions at several points. Importance or radiance distribution from particles at a particular point in a scene is modeled by a GMM and parameters of the model are obtained by the maximum a-posteriori(MAP) technique where prior model is combined with GMM which is incrementally learned by means of the expectation maximization(EM). In this paper, the whole training process is exactly same with the [2]. We focus on recover the training process and use the 2D image as the example to improve the regression model. Our key contributions are:

1 Generate the possibility distribution function(PDF) proportional with the intensity of the 2D image.
2 Introduction of on-line learning of parametric mixture models to image simulation.
3 Strategies for improvements of the image simulation parametric mixture models.

II. Our Approach

In this paper, we take the Lenna image as the 2D object to train this parametric mixture model. The Fig.1 show the sparse version of the approach. We firstly sampled from the image proportional with its intensity. An initial estimate of their parameters is computed and iteratively improved using expectation maximization in the off-line learning process. Finally, after the model get converged in the on-line learning process, the parametric mixture model is transformed to the image representation. Because we use the same learning algorithm as the paper [2], all the basic algorithms can be found in its background section, including the classic batch EM(Expectation Maximization) algorithm, the off-line stepwise EM algorithm[4] and the deriving of the on-line stepwise EM algorithm. And because both stepwise EM variants are essential components in our method,[3] and [5] provide more details on EM. In this section, we will focus on the steps we listed before excepted the learning algorithms.

A. Infinite Pixels Sampling Proportional to the Intensity

In order to generate infinite samples from the image based on the pixel intensity and then train it using the on-line GMM model, it is necessary to be able to transform the pixel intensity to probability distribution. This section will introduce how to get the probability distribution and use the inversion method to get the random samples proportional to the intensity distribution.

In the image, the intensity of each pixel is related with location information. We use the 2D location information to represent a pixel and the amount of samples generated in each location will be proportional to its intensity. To generate samples proportional with the intensity of pixels, we need to construct the 2D distributions firstly and then use the inversion method[6] to generate samples based on that distribution.

In our case, the intensity of each pixel is the value of the 2D function, and the location(x, y) is the input parameters of the function.Because the intensity of each pixel is discrete values, [6] introduce the method for generating samples from a piecewise-constant 2D distribution. Followed up by that algorithm, we need to compute the average value of each

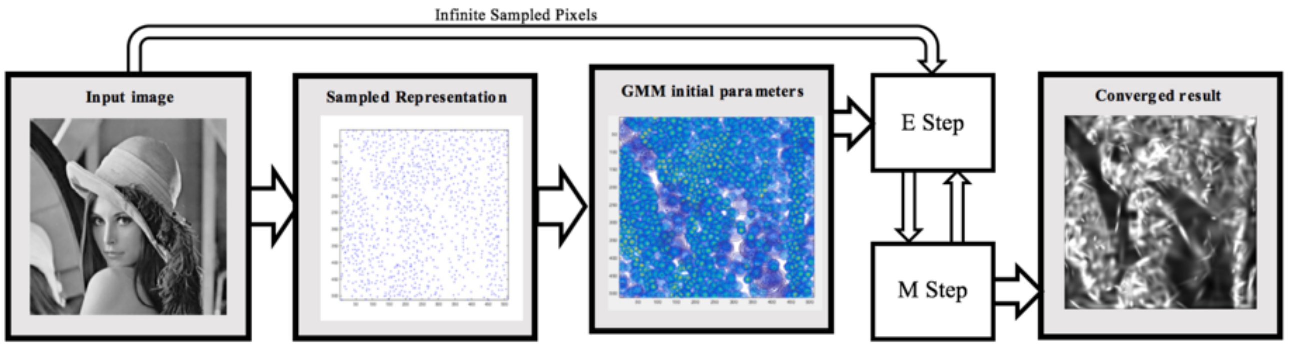
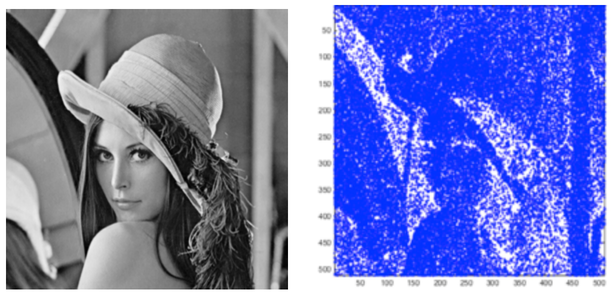


Fig. 1: Our Approach

row $A(x)$, and compute the Cumulative Distribution Function(CDF) of $A(X)$. Then use the inversion method to select one specific row based on the CDF. In this way,we have decided which row will be chosen to generate the points. Then the CDF $P_r(x)$of this row should has been generated in the first. We just need to use the inversion method again to pick a y based on the $P_r(x)$.



(a) Original Lenna Image (b) Samples base on Intensity

Fig. 2: Sampling Proportional with Intensity

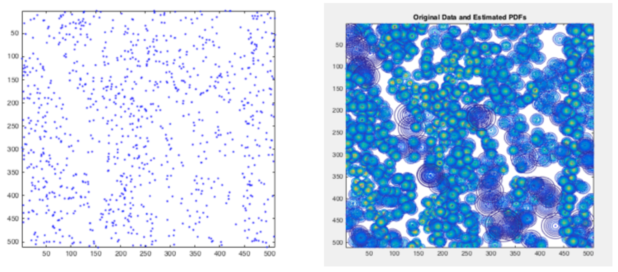
So we use these methods to generate the PDF and CDF based on the intensity values of the image and then sampling from the pixels. And because the pixels of image are not infinite, we just sample from the image in a infinite loop until the training result converged. This process will also be introduced in the section II C. Fig.2a is the original Lenna image (500× 500), and Fig. 2bshows the 15000 random samples generated adapt to the image intensity.

B. GMM Initialization

As the algorithm in paper [2], we will use the MAP method to train the GMM model. It is important to note that while MAP is guaranteed to converge, it will generally only find a locally optimal solution. Hence it is important to supply it with a reasonable initial parameter estimate.

The original way to set the initial parameters are using random strategy, which means we will randomly generate K(the amount of composite Gausses in GMM) samples, and use their location to be the initial μ of GMM. Set the

(a) Original Lenna Image



(b) Initialized Samples (c) Initialized GMM

Fig. 3: Initialization

covariance of all the samples to be the initial covariance of each Gauss. For the weight, we will set it to be equivalent with each component. This initialization method may work well for the sampler clustering problem. If we want to train the GMM to simulate an image instead of clustering 4 or 5 classes, the initialization should be much more approximate with the original model, otherwise the converging process will be very slow, and even the GMM has been converged, it's not the optimized result. When we use the GMM model consists of just 50 components, the converged gaussian contours have much space overlapped, which make the final image so fuzzy. If we use more component in the model, and still initialized with the traditional method, there maybe much more component overlapped with each other. From the [7], we learned that if we want to get a more accurate final image quickly, we should set the initial parameters of the GMM also based on the image intensity, which means the darker place should have fewer gausses and the brighter place should have more gausses, and finally all the gausses in the GMM filling up the whole image. Using this strategy, the most apparent change compared with the original one is that we need to set different covariance of each component instead of set the covariance of all the samples to be the initial value.

The PDF of the image adapt to its intensity is generated in the last section. Then the sigma can be computed as following:

$$pdf(x) = \frac{1}{2\pi|\Sigma|^{\frac{1}{2}}} e^{(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))}$$

Given that we have know the $pdf(x)$, and $x = \mu$, we can compute the Σ from this equation directly.

$$|\Sigma| = (\frac{1}{2\pi pdf(x)})^2$$

and because we assume

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

then we can get

$$\sigma^4 = (\frac{1}{2\pi pdf(x)})^2,$$

so

$$\sigma^2 = \frac{1}{2\pi pdf(x)}.$$

Finally, normalize the sigma to be $\sigma^2 = \frac{1}{2\pi pdf(x)k}$, k is the number of gaussians.The covariance are set based on this function. The Fig. 3c show the initialized GMM result based on the Lenna image.

C. *Learning Distributions from Image Samples*

The pixels sampling algorithm generates samples $s_i = (x_i, y_i)$, defined by their position x_i, y_i. And an approximation of a sampling PDF p has been reconstructed from the intensity of image. So we will learn the distributions from the image samples. And the algorithms we will use in this process, the stepwise EM algorithms(both off-line and on-line), have been presented in [2] that supports the density estimation.

In the first part of the learning process, we firstly generate some initial samples to initialize the GMM and use the off-line learning to train the parameters in GMM until it converged. Then in the second part, the parameters in the converged GMM we get from the first training part is used as the start parameters of the on-line stepwise EM training process. Because the pixels of image are not infinite, we just make samples from the image in a infinite loop until the on-line training result converged.

The equations of log-likelihood and stepwise EM E-step are all given in the section 4.2 of paper [2], and because we do not take the weight of each pixel into consideration, so here we will give their derived function for model parameters without the weight, or all the weights could be considered as 1. $\theta^{new} = \{\pi_1, \mu_1, \sum_1, ..., \pi_K, \mu_K, \sum_K\}$. The specific formulae defining the vector function θ read as follows:

$$\pi_j^{new} = \frac{(\mu_\gamma)_i^j + \frac{v-1}{n}}{1 + \frac{K(v-1)}{n}}, \tag{1}$$

$$\mu_j^{new} = \frac{(s)_i^j}{(\mu_\gamma)_i^j} \tag{2}$$

$$\Sigma_j^{new} = \frac{\frac{b}{n}I + (ss^\top)_i^j - A + (\mu_\gamma)_i^j B}{\frac{a-2}{n} + (\mu_\gamma)_i^j} \tag{3}$$

where

$$A = (s)_i^j(\mu_j^{new})^\top + \mu_j^{new}(s^\top)_i^j$$
$$B = \mu_j^{new}(\mu_j^{new})^\top,$$

Despite the part of the weights, the representation and the scalars setting are all same with the [2] as following. I is the identity matrix, K is the number of mixture components and n is the total number of observed samples (see the details below). Scalars a, b and v are parameters of conjugate priors induced by the MAP solution (see [8] for more details). In our implementation, we also use the value $a = 2.01, b = 5 \times 10^{-4}, v = 1.01$.

D. *Transformation from GMM Parameters to Image*

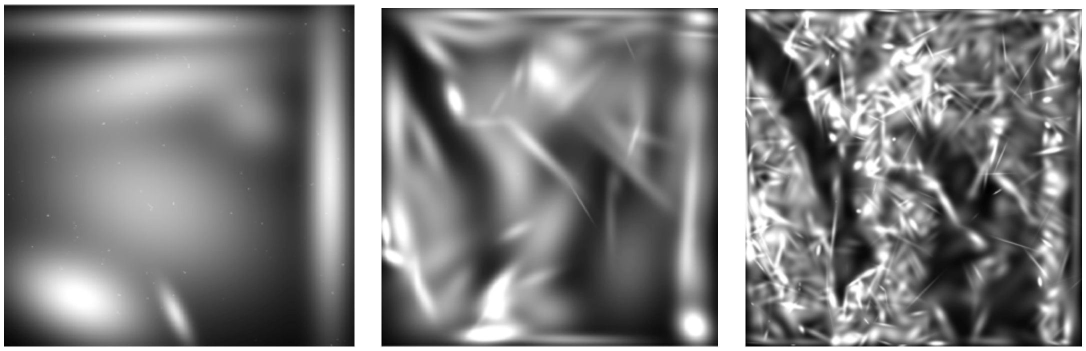
After the GMM converged, we need to transform it to the result image. Because in the section III A, the PDF is generated proportional to the intensity, the transformation from GMM to image is actually the inverse process. Firstly, we need to compute discrete pdf p_i^j for each location, and then sum them up to be $sum\_p_i^j$ for generalization. So the function of computing final image intensity is: $\frac{p_i^j}{sum\_p_i^j} \times sum\_int$.

III. Experiment Results and Optimization

From the section III, we have introduced the whole approach. This section we will give the on-line learning results and some method we used to optimize the results.

The number of observed samples n in Equations (6) and (8) governs the effect of our prior beliefs. The more samples we have observed the weaker the effect of priors. In on-line stepwise EM, we simply set n to the current value of the index of sufficient statistics i. However, to fully exploit the MAP approach and thus to prevent over-fitting in our off-line stepwise EM, it is necessary to set $n = min(i, N)$. This is necessary because the algorithm iterates over the same batch of N samples multiple times before it converges and the index i could be much higher than the actual number of unique observed samples. From the experiments, we found that both the on-line and the off-line stepwise EM algorithms achieve the best results when the M-step is executed every m = 10 times of K samples (see Alg. 1) and with the stepsize parameter $\alpha = 0.7$. The Fig.4 shows the resulted image that with different proportion between K and m. It's apparent that when m is 10 times of K, the result is more accurate, and when m is smaller than K, the training of GMM will be overfitting.

Even though, the result image with the coordinate proportion between K and m could have represented the image, but



(a) K=1000,m=100 (b) K=1000,m=1000 (c) K=1000,m=10000

Fig. 4: Coverged Result with Different Ratio between K and m

it's still not that clear. In the following sections, we try to use some other method to optimize the training process and get the more accurate result image.

A. Stratified sampling method

From the Fig.3b and Fig.3c, we can see that if we use the independent random sampling method, there will be some points so close or so far to each other, which result in the initial gauss contour overlapped with each other, at the same time, some space of the image even is not covered. Because the EM method is much sensitive with the initial parameters, we decided to use another sampling strategy to overcome this disadvantage.

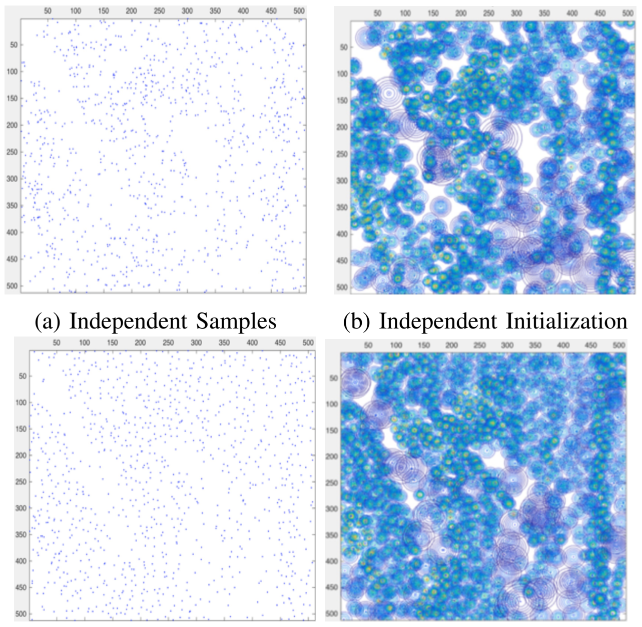
For the stratified sampling method, when you want to generate n stratified points in a space, the much easier way to understand it is that these n points should be sampled equally on a $\sqrt{n} \times \sqrt{n}$ grid. And the biggest difference between stratified sampling and random sampling is just the part of generate random value, the part that generate points proportional with the intensity is still same. So what we do is firstly set the grid, and then traverse each cell in the grid to generate the random points. The Fig.5 show the comparison between the random and stratified sampling strategy.

From the Fig.5b and Fig.5d, we can see that by using the stratified samples, the image has been filled better than the independent samples. And then we will test if this will be helpful for the convergence process. We use the GMM initialized with K = 100, m = 1000 to do this experiment, and the other conditions are all set with the same value. The Fig.6 show the comparison between these two sampling method.

It's apparent to see that the resulted image with stratified samples are much accurate than the original one.

B. Reduce the Skinny Gausses in GMM

From the Fig.4c K = 1000, m = 10000, we can see that there are many skinny Gausses in the image, which make the image much difficult to recognize. So we want to make some constraint of the Gausses in the convergence process, which may reduce the skinny gausses in the end.

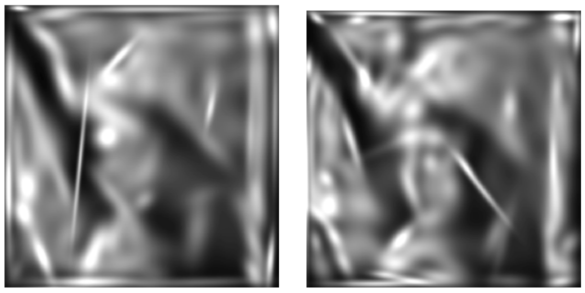


(a) Independent Samples (b) Independent Initialization

(c) Stratified Samples (d) Stratified Initialization

Fig. 5: Pictures of animals

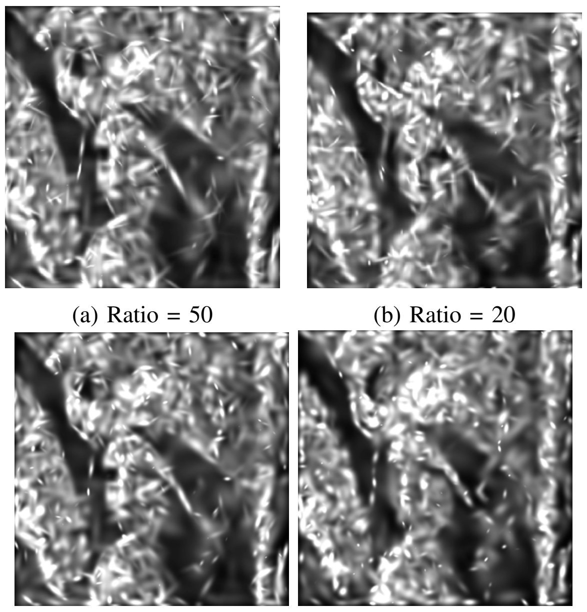
1) SVD Method: Firstly, we use the Singular value decomposition method to decompose the covariance matrix $Sigma$. So the $Sigma$= $U\Sigma V$, Σ_{11} and Σ_{22} are the singular values of Sigma. Because we just want to adjust the ratio between the singular values, the product of them would not be changed. After the ratio between the singular values are adjusted, we will recompute the Sigma with the new singular values. In this way, we could set the constraint between the two orthorhombic directions in each gauss component. If the ratio between the two directions are too large, this gauss will become more skinny, in the opposite, when the ratio between the two direction is much close to 1, the gauss will become more similar with a circle. Fig.7 shows the different result images with the different constraint. And we can see that the smaller the constraint factor is the less skinny gausses will appear. The result image with less skinny gausses are more clear than the



(a) Independent (b) Stratified

Fig. 6: Comparison between these Two Sampling Strategy

original one.



(a) Ratio = 50 (b) Ratio = 20

(c) Ratio = 10 (d) Ratio = 5

Fig. 7: SVD Constraint Result

2) Scalar Covariance Update Function: Even though we have used the SVD method to reduce the skinny gausses in the result image, it's still a constraint method. There will be a concern about whether this constraint process will affect the training process. Because the constraint method is actually reducing the radio between the two dimensions, so what if the covariance matrix is updated in the form of a scalar value, where the ratio will be exact 1. Finally, we derived the scalar covariance update function from the EM method we introduced in the section III C. The derivation process of using scalar sigma in EM to update GMM is as following.

a) Derivation for Batch EM: Assume the GMM composed of K gaussian components, the pdf of the GMM($\theta = (\pi_k, \mu_k, \Sigma_k)$) is:

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|\mu_k, \Sigma_k) \tag{4}$$

So the Likelihood Function of the GMM(N = the size of the dataset) should be:

$$\prod_{i=1}^{N} p(x_i) = \prod_{i=1}^{N} \{\sum_{k=1}^{K} \pi_k p(x_i|\mu_k, \Sigma_k)\} \tag{5}$$

Because the possibility of each point is usually a very small number, and the production of many small numbers will cause the underflow issue with floating point numbers, we will take its log to transform the production function to a summation function and get the log-likelihood function as following:

$$\sum_{i=1}^{N} \log\{\sum_{k=1}^{K} \pi_k p(x_i|\mu_k, \Sigma_k)\} \tag{6}$$

Because the equation(9) $p(x)$ will be expanded as following:

$$p(x) = \sum_{k=1}^{K} \frac{\pi_k}{\sqrt{(2\pi)^d |\Sigma_k|}} e^{-\frac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)} \tag{7}$$

And the Σ_k is changed to be a scalar parameter:

$$\Sigma_k = \sigma_k^2$$
$$\Rightarrow |\Sigma_k| = (\sigma^2)^d$$
$$\Rightarrow \Sigma_k^{-1} = \frac{1}{\sigma_k^2}$$

Then the function becomes:

$$p(x) = \sum_{k=1}^{m} \frac{\pi_k}{\sqrt{(2\pi\sigma_k^2)^d}} e^{-\frac{1}{2}\frac{(x-\mu_k)^\top (x-\mu_k)}{\sigma_k^2}} \tag{8}$$

So the parameters in Gauss θ becomes: $\theta = (\pi_k, \mu_k, \sigma_k^2)$ and the $h_k = \sigma_k^2$ can be used to make $\theta = (\pi_k, \mu_k, h_k)$.
Because the equation(11) need to be maximized, however, the $\log \Sigma$ is a challenge when doing the maximization. The normal way to do that is to use Jensen Inequality as following, and the log-likelihood function becomes (z is the latent variable):
$\sum_{i=1}^{N} \log \sum_{k=1}^{K} p(x_i) = \sum_{i=1}^{N} \log\{\frac{\sum_{k=1}^{K} p(x_i, z_i = k|\theta)}{p(z_i = k|\theta_t)} p(z_i = k|\theta_t)$

$$\geq \sum_{i=1}^{N} \sum_{k=1}^{K} p(z_i = k|\theta_t) \log\{\frac{p(x_i, z_i = k|\theta)}{p(z_i = k|\theta_t)}\} \tag{9}$$

Represent $\gamma_{ik} = p(z_i = k|\theta_t)$ (γ_{ik} is the possibility that data x_i belonged to the kth Gauss; θ_t is the updated parameter after the current E-step). Now equation (14) just need to be maximized, which has is expanded as the following $L(\theta)$

$$L(\theta) = \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{jk} \left[\log \pi_k - \frac{d}{2} \log 2\pi h_k - \frac{1}{2} \frac{(x_i - \mu_k)^\top (x_i - \mu_k)}{h_k}\right] \tag{10}$$

And because there is a constraint that $\sum_{k=1}^{K} \pi_k = 1$, I use Lagrange multiplier to add this constraint in the maximization process as following $T(\theta)$.

$$T(\theta) = L(\theta) + \lambda(\sum_{k=1}^{K} \pi_k - 1) \tag{11}$$

Maximization:
Get the derivation and set it to equal to zero.Then the parameters θ become:
$\frac{\partial T}{\partial \pi_k} = \sum_{i=1}^{N} \frac{\gamma_{ik}}{\pi_k} + \lambda = 0$

$$\Rightarrow \pi_k = \frac{\sum_{i=1}^{N} \gamma_{ik}}{\sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik}} \tag{12}$$

$\frac{\partial T}{\partial \mu_k} = \sum_{i=1}^{N} \gamma_{ik} \frac{x_i - \mu_k}{h_k} = 0$

$$\Rightarrow \mu_k = \frac{\sum_{i=1}^{N} \gamma_{ik} x_i}{\sum_{i=1}^{N} \gamma_{ik}} \tag{13}$$

$\frac{\partial \pi}{\partial h_k} = -\sum_{i=1}^{N} \gamma_{ik} \frac{d}{2} \frac{1}{h_k} + \frac{1}{2} \sum_{i=1}^{N} \gamma_{ik} \frac{(x_i - \mu_k)^\top (x_i - \mu_k)}{h_k^2} = 0$

$$\Rightarrow h_k = \frac{\sum_{i=1}^{N} \gamma_{ik} (x_i - \mu_k)^\top (x_i - \mu_k)}{d \sum_{i=1}^{N} \gamma_{ik}} \tag{14}$$

Finally, deviation result with $\Sigma_k = h_k I$ is as following, and the two Steps in EM become:

E-Step: $\gamma_i k = p(z^j = k|\theta^t)$

M-Step:

$$\pi_k = \frac{\sum_{i=1}^{N} \gamma_{ik}}{\sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik}} \tag{15}$$

$$\mu_k = \frac{\sum_{i=1}^{N} \gamma_{ik} x_i}{\sum_{i=1}^{N} \gamma_{ik}} \tag{16}$$

$$h_k = \frac{\sum_{i=1}^{N} \gamma_{ik} (x_i - \mu_k)^\top (x_i - \mu_k)}{d \sum_{i=1}^{N} \gamma_{ik}} \tag{17}$$

In our case, dimension d = 2.

b) Derivation for Online EM[8]: Given the Gauvain's update formula for the matrix Σ_j of j-th Gaussian in the mixture reads:

$$\Sigma_j^{''} = \frac{b_j I + \Sigma_{q=0}^{N-1} \gamma_{qj} (s_q - \mu_j)(s_q - \mu_j)^\top}{(a_j - 2) + \Sigma_{q=0}^{N-1} \gamma_{qj}} \tag{18}$$

Because in the equation(23), scalars a,b and v are parameters of conjugate priors induced by the MAP solution [2]. I is the identity matrix. $a > d - 1$, $b > 0$ are hyper-parameters, and d = 2 is the dimension. [3] provides details on the use of Dirichlet and Wishart distributions as conjugate priors.
So we use the same conjugate as the on-line paper [2] and by using the derivation result of h_k as above, the Σ_j could be generated as following:

$$\Sigma_j^{'} = \frac{b_j I + \Sigma_{q=0}^{N-1} \gamma_{qj} (s_q - \mu_j)^\top (s_q - \mu_j)}{(a_j - 2) + d \Sigma_{q=0}^{N-1} \gamma_{qj}} \tag{19}$$

By using simple algebra, we get:

$$(s_q - \mu_j)^\top (s_q - \mu_j) = s_q^\top s_q - s_q^\top \mu_j - \mu_j^\top s_q + \mu_j \mu_j^\top \tag{20}$$

Substituting (25) into (24) and multiplying both the nominator and denominator by $\frac{1}{N}$, the formula for Σ_j becomes:

$$\Sigma_j^{'} = \Sigma_j^{'} \frac{\frac{1}{N}}{\frac{1}{N}} = \frac{\frac{b_j}{N} + \frac{1}{N} \Sigma_{q=0}^{N-1} \gamma_{qj} s_q^\top s_q - A + B}{\frac{(a_j - 2)}{N} + \Sigma_{q=0}^{N-1} \frac{\gamma_{qj}}{N}} \tag{21}$$

where,

$$A = \frac{1}{N} \Sigma_{q=0}^{N-1} \gamma_{qj} s_q^\top \mu_j + \frac{1}{N} \Sigma_{q=0}^{N-1} \gamma_{qj} \mu_j^\top s_q$$
$$B = \frac{1}{N} \Sigma_{q=0}^{N-1} \gamma_{qj} \mu_j^\top \mu_j$$

Then the fact to obtain MAP update formula[1] of the covariance matrix(the sufficient statistics: a triplet $u_i^j = ((u_\gamma)_i^j, (s)_i^j, (ss^\top)_i^j)$)is:

$$\Sigma_j^{'} = \frac{\frac{b_j I}{N} + (s^\top s)_i^j - A + (u_\gamma)_i^j B}{\frac{(a_j - 2)}{N} + d (u_\gamma)_i^j} \tag{22}$$

where,

$$A = (s^\top)_i^j \mu_j + \mu_j^\top s_i^j$$
$$B = \mu_j^\top \mu_j$$

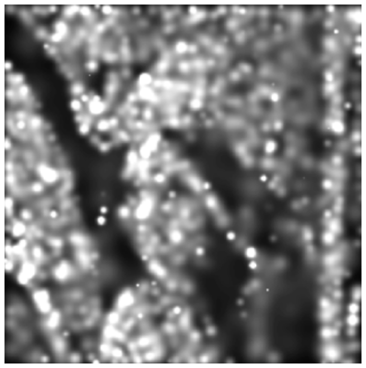


Fig. 8: Scalar Covariance Result Image

After training based on the scalar covariance update function, we got the result image as Fig.8 we can see that this time, all the Gausses in GMM are much more like a circle, and no skinny gausses will show up again. But when compare the scalar covariance constraint method with the SVD constraint method, the SVD result with smaller factor are more accurate since the result images are more clear and more similar with the true image.

IV. Conclusion

We have proposed the use of a parametric mixture model to represent an image. The core of our approach is an online learning procedure that allows one to train the distributions from a potentially infinite stream of samples. With this approach, we can represent any complex image using a parametric mixture model, and it will be helpful for recovering good importance sampling distributions in difficult and complex lighting configurations, where an excessively large number of samples would otherwise be necessary. And in the future, this approach will be used in more areas.

References

[1] H. W. Jensen, “Importance driven path tracing using the photon map,” in *Rendering Techniques 95*. Springer, 1995, pp. 326–335.

[2] J. Vorba, O. Karlík, M. Šik, T. Ritschel, and J. Křivánek, “On-line learning of parametric mixture models for light transport simulation,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 101, 2014.

[3] C. M. Bishop, “Pattern recognition,” *Machine Learning*, vol. 128, 2006.

[4] P. Liang and D. Klein, “Online em for unsupervised models,” in *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*. Association for Computational Linguistics, 2009, pp. 611–619.

[5] O. Cappé, “Online expectation-maximisation,” *Mixtures: Estimation and Applications*, pp. 1–53, 2011.

[6] M. Pharr and G. Humphreys, *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2004.

[7] M. Papas, W. Jarosz, W. Jakob, S. Rusinkiewicz, W. Matusik, and T. Weyrich, “Goal-based caustics,” in *Computer Graphics Forum*, vol. 30, no. 2. Wiley Online Library, 2011, pp. 503–511.

[8] J. Vorba, O. Karlık, M. Šik, T. Ritschel, and J. Krivánek, “On-line learning of parametric mixture models for light transport simulation–supplemental material.”