Learning High-order Generative Texture Models

Ralph Versteegen, Georgy Gimel'farb and Patricia Riddle
Department of Computer Science
The University of Auckland
Auckland 1142, New Zealand
rver017@aucklanduni.ac.nz, {g.gimelfarb, p.riddle}@auckland.ac.nz

ABSTRACT

We introduce a new simple framework for texture modelling with Markov–Gibbs random fields (MGRF). The framework learns texture-specific high order pixel interactions described by feature functions of signal patterns. Currently, modelling of high order interactions is almost exclusively achieved by linear filtering. Instead we investigate 'binary pattern' (BP) features which are faster to compute and describe quite different properties than linear filters. The features are similar to local binary patterns (LBPs) — previously not applied as MGRF features — but with learnt shapes. In contrast to the majority of MGRF models the set of features used is learnt from the training data and is heterogeneous. This paper shows how these features can be efficiently selected by nesting the models. Each new layer corrects errors of the previous model while allowing incremental composition of the features, and uses validation data to decide the stopping point. The framework also reduces overfitting and speeds learning due to a feasible number of free parameters to be learnt at each step. Texture synthesis results of the proposed texture models were quantitatively evaluated by a panel of observers, showing higher order BP features resulted in significant improvements on regularly and irregularly structured textures.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]; I.2.10 [**Vision and Scene Understanding**]: Texture; I.4 [**Image Processing and Computer Vision**]; I.4.7 [**Feature Measurement**]: Texture

General Terms

Theory; algorithms; experimentation

1. INTRODUCTION

Texture modelling is important to many computer vision and image processing tasks such as image segmentation, inpainting, classification, synthesis, anomaly (defect) detection, and so forth. Although successful specialised algorithms for solving these problems have been developed, generative probabilistic models which provide explicit iamge probabilities for a specific texture class remain appealing. These models may be applied not only to all of the above tasks, but also to others where appearance priors or feature extraction are needed, and they are of interest to understanding human vision. Unlike discriminative models, generative models need to capture texture features that are significant to human perception in order to be successful.

The most prevalent tool for image and texture modelling are Markovian undirected graphical models, a.k.a. Markov random fields (MRFs). A Markov–Gibbs random field (MGRF) is an MRF specified with an explicit Gibbs probability distribution (GPD). A GPD is factorised over complete subgraphs (cliques) of an interaction graph with variables/pixels as nodes. which Factors quantify the strength of interactions between those variables and can be described in terms of feature functions which identify certain signal configurations (patterns), each of which has a corresponding weight/parameter. The order of a feature is its arity, and in general higher order one cannot be decomposed into lower order features. MGRFs and other statistical texture models reduce images to a vector of statistics of image features, which are assumed sufficient to describe the texture. Historically MRF models used statistics of pairs of pixels [3, 5]. However higher-order MGRFs have become more common as they are recognised to be necessary for more expressive models of natural images and textures, by abstracting beyond pixels [7, 18, 25]. In addition, since regularly tiled textures have strong long range correlations between nearby tiles it is natural to learn an interaction structure specific to the texture. Yet it is still almost unheard of in computer vision and image modelling for higher-order MRF structure to be learnt rather than hand selected.

The higher-order MGRFs in use nearly exclusively apply linear filters as feature functions, with statistics of the filter responses, such as means and covariances [16] or histograms [25], forming a description vector. More general high order features require specification of clique shapes (e.g. circular rings for local binary patterns (LBPs) [14, 15]) and a reasonably parameterised functional form from amongst a combinatorially huge space. In order to explore this neglected possibility, this paper describes a model nesting procedure which greedily adds features to a model to correct statistical differences between the training data and current

model, and attempts to build higher order features by composing lower order ones. Texture synthesis is used as a case study. Validation testing is used to reduce overfitting by stopping the procedure when additions to the model worsen it, avoiding an arbitrary stopping criterion.

The contributions of this paper are four-fold: (*i*) A method of efficiently selecting both high order features and parameters by rapidly nesting models is introduced. The proposed models use no hidden variables, which eases learning and inference. (*ii*) High order 'binary pattern' (BP) texture features generalising and extending the very popular LBP image descriptors are used in MGRFs. These features are quite different from the mainstream high order linear filtering or Potts potentials, and much faster to compute than responses of linear filters. Even traditional LBPs have apparently never been used in this way. (*iii*) Several schemes for selecting features of up to 9th order are compared. (*iv*) The ability to learn characteristic features across different types of textures is explored with texture synthesis, evaluated by a panel of observers. The paper is structured as follows: Section 2 overviews related prior work; Section 3 gives the theoretical foundation; Section 4 describes the nesting algorithm and implementation details; experimental results are presented in Section 5.

2. RELATED WORK

High order MGRF models. Research in texture analysis has focused on describing textures using the distributions of responses of square linear filters. MGRF texture models utilising filters were introduced with FRAME [25], where the filters were selected from a manually-specified bank. More recently learning the filters themselves has been popularised by the Field-of-Experts (FoE) model [18]. This means that the interaction structure is learnt implicitly (via filter coefficients). FoE was successfully extended and applied to texture modelling by Heess et al.'s [7] bimodal FoE (BiFoE); several state of the art generative texture models have been built on BiFoE, some using various configurations of hidden variables. Kivinen and Williams [9] improved on BiFoE by using gated MRFs [17], and Luo et al. [12] investigated convolutional deep belief networks (DBN). However because of learning difficulties and to make required computation feasible all these learnt-filter models have been restricted to relatively small filter sizes. These do not directly capture distant interactions; the largest size used in the mentioned works was 11×11.

Structure Learning. An intuitive method for selecting the features of the model is to use greedy heuristic sequential selection, which has been proposed and used by a number of different authors (eg. [4, 23, 25]) in slightly different forms. This alternates between adding one or more features/factors to the current model from a candidate set according to which has the largest 'error', in some sense, between the training data and model [23, 25], and then finding the new maximum likelihood estimate (MLE) of the parameters. This paper describes a very similar model nesting approach, however parameters corresponding to previous features are kept fixed and an attempt is made merely to improve on the base model rather than search for the MLE.

Della Pietra et al. [4] treated structure learning in MGRFs (for modelling natural language) as a feature selection problem, and built up higher order features out of lower order ones. Several other authors have since considered selecting MGRFs features by gradually composing together low order atomic features (e.g. [19]), or by starting from template-like features which are conjunctions of simple $x_i = c_j$ predicates, and gradually generalising them (e.g. [13]). A closely related popular method for MGRF structure learning is the use of sparsity-inducing L_1 regularisation [11], which forces some feature weights to exactly zero so that they can be removed. This has the advantages that the regularisation combats overfitting and that it allows removing a feature after it has been added.

Texture synthesis. Practical texture synthesis is currently dominated by algorithms which combine pieces (pixels or patches) of a source image so that the pieces fit together well, defined by the match between neighbourhoods of the pieces (e.g. [10, 22]). These can produce excellent results and are fast, but are not texture models and do not provide descriptions of texture. Another rather successful but far less efficient class of texture synthesis algorithms is based on generating images with certain statistics equal to those of the training image, in particular statistics of wavelet-decompositions [16]. These are implicit MRF texture models as they provide no explicit probability distributions and hence are less broadly applicable than probabilistic models.

3. NESTED RANDOM FIELDS

In this section we first define MGRFs and their application to texture modelling, and then show how general exponential distributions as the conceptual and theoretical basis for model nesting.

3.1 MGRF image models

Let $\mathcal{R} \subset \mathbb{Z}^2$ be a finite lattice of integer image coordinates. This work is restricted to translation-invariant MGRFs, that is, the factors and cliques (sets of mutually interacting pixels in $\mathcal{R}$) are repeated across $\mathcal{R}$ by translation. An order d clique family in $\mathcal{R}$ is the set of all possible cliques which are translations of a single template (that is, a clique shape), given as a list of d coordinate offsets $\boldsymbol{\alpha} = \alpha_1, \ldots, \alpha_d$, ($\alpha_i \in \mathbb{Z}^2$) between the pixels of the clique, with $\alpha_1 = (0, 0)$ fixed. Let $\mathbf{G} : \mathcal{R} \to \{0, \ldots, Q-1\}$ be an image on $\mathcal{R}$ with Q possible grey levels. An order k feature is a function $f : \{0, \ldots, Q-1\}^k \to \mathbb{N}_s$ with finite range $\mathbb{N}_s := \{1, \ldots, s\}$ with an associated clique family C_f given by offset list $\boldsymbol{\alpha}_f$. The histogram of values of f collected over an image $\mathbf{G}$ is the vector

$$\mathbf{h}_f(\mathbf{G}) = \left[\sum_{c \in C_f} [\![l = f(\mathbf{G}_c)]\!] \ : \ l \in \mathbb{N}_s \right] \tag{1}$$

where $[\![\cdot]\!]$ is the Iverson bracket mapping true $\mapsto 1$, false $\mapsto 0$, and $\mathbf{G}_c$ is the sequence of pixels $\mathbf{G}(r_1), \ldots, \mathbf{G}(r_d)$, $r_i \in c$.

A homogeneous MGRF model p of images is a Gibbs probability distribution over images on $\mathcal{R}$ given by [8]

$$p(\mathbf{G}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(-\mathbf{f}(\mathbf{G}) \cdot \boldsymbol{\theta}) \tag{2}$$

which is an exponential family distribution with sufficient statistics being the feature vector $\mathbf{f}(\mathbf{G})$ (a concatenation of histograms for a set F of feature functions: $\mathbf{f}(\mathbf{G}) := [\mathbf{h}_f(\mathbf{G}) : f \in F]$) and a vector of parameters $\boldsymbol{\theta}$ (a concatenation of per-feature parameter vectors $\boldsymbol{\theta} = [\boldsymbol{\theta}_f : f \in F]$). $Z(\boldsymbol{\theta})$ is a normalisation constant. The order of the model is defined as the maximum order of any of its features.

Given F and a training image $\mathbf{G}_{\text{obs}}$ the maximum likelihood estimate (MLE) of the MGRF parameters $\boldsymbol{\theta}^* := \arg\max_{\boldsymbol{\theta}} p(\mathbf{G}_{\text{obs}}|\boldsymbol{\theta})$ gives a model such that the expected values of $\mathbf{f}(\mathbf{G})$ under p are equal to $\mathbf{f}(\mathbf{G}_{\text{obs}})$ (the sufficient statistics). This can be seen from the log-likelihood gradient:

$$\nabla \log p(\boldsymbol{\theta}_i|\mathbf{G}_{\text{obs}}) = \mathbb{E}_{p_i}[\mathbf{f}_i(\mathbf{G})] - \mathbf{f}_i(\mathbf{G}_{\text{obs}}). \quad (3)$$

3.2 Model nesting

In a sequential structure selection procedure, features are gradually added to a model based on the disagreement between their sufficient statistics (i.e. histograms) in the training image and the model's expected statistics, in order to correct those errors. Let the feature function vector $\mathbf{f}_{i+1}(\mathbf{G})$ be the new features to be added to the current model p_i. Suppose one has constraints to be satisfied by a model p_{i+1} in the form of expectations $\mathbb{E}_{p_{i+1}}[\mathbf{f}_{i+1}(\mathbf{G})] = \mathbf{f}_{i+1}(\mathbf{G}_{\text{obs}})$, and already has prior information expressed as a base model (in this case the model at the previous iteration, p_i). It is widely known [4] that the model p_{i+1} which matches the new constraints but deviates from the base model the minimum possible amount (as measured by the Kullback-Leibler divergence) has a simple form. It is a maximum entropy (general exponential family) distribution

$$p_{i+1}(\mathbf{G}|\boldsymbol{\theta}_{i+1}) = \frac{1}{Z(\boldsymbol{\theta}_{i+1})} p_i(\mathbf{G}) \exp(-\mathbf{f}_{i+1}(\mathbf{G}) \cdot \boldsymbol{\theta}_{i+1}) \quad (4)$$

where the parameters $\boldsymbol{\theta}_{i+1}$ are chosen to maximise the data likelihood. Note that there is no need to modify the parameters of the base model, though in other works this is normally done, regarding the new model as an exponential distribution with combined features $\mathbf{f}_1, \ldots, \mathbf{f}_{i+1}$.

4. SEARCH PROCEDURE

We greedily add features to the model which have histograms with maximum disagreement with the training image. There are two significant differences to the traditional sequential algorithm. Firstly, here only the parameters associated with $\mathbf{f}_{i+1}$ are optimised rather than all of them, in order to reduce overfitting. Secondly, additions are reverted if comparison to validation data repeatedly shows worse scores, providing the stopping condition. A generic intrinsic validation score $V(p_i|F_i)$ not specific to texture modelling was used, as described in Section 4.1. Waiting multiple iterations before stopping is necessary because the approximate samples rapidly drawn from the model (see Section 4.2) have high variability and error in their statistics.

We perform feature selection based on the assumption that when some configuration of pixels in a texture is characteristically common, that the configuration restricted to fewer pixels should also be common. Given a set of feature functions F_i which have already been selected, a *selector* function $C(F_i)$ provides a candidate set of new features, possibly built upon previous ones, allowing the gradual building up of features from pieces, while reducing the set of candidate features to a size that can be exhaustively searched at each iteration. The algorithm proceeds through a sequence of selectors $C^1, \ldots, C^k$, each providing features of a certain order and type, to avoid the problem of comparing across feature types.

Require: Initial base MGRF model p_0 with features F_0, training image $\mathbf{G}_{\text{obs}}$
  Set current best model $p^* \leftarrow p_0$
  for C in $C^1, \ldots, C^k$ **do**
    loop
      Obtain set of samples $\{\mathbf{G}_j^{(i)}\}_{1 \le j \le k}$ from model p_i
      for each $f \in C(F)$ **do**
        Collect $\mathbf{h}_f(\mathbf{G}_{\text{obs}})$ and $\mathbf{h}_f(\mathbf{G}^{(i)}) := \frac{1}{k}\sum_j \mathbf{h}_f(\mathbf{G}_j^{(i)})$
      end for
      Pick one or more f with maximal $D_{\text{JS}}(\mathbf{h}_f(\mathbf{G}_{\text{obs}}), \mathbf{h}_f(\mathbf{G}^{(i)}))$ {see Section 4.1}
      $F_{i+1} \leftarrow F_i \cup \{f\}$
      if $V(p_i|F_{i+1}) > V(p^*|F_{i+1})$ **then**
        $p^* \leftarrow p_i$
      else if $p_{i-3}, \ldots, p_i$ have all scored worse than p^* **then**
        $p_{i+1} \leftarrow p^*$ {revert to best model}
        Break to the outermost loop
      end if
      Form p_{i+1} by adding f to p_i and learning parameters $\boldsymbol{\theta}_{i+1}$ as in Eq. (4)
    end loop
  end for

4.1 Validation

For comparing probability distributions p and q we used the Jensen–Shannon divergence (JSD) $D_{\text{JS}}(p \parallel q)$, which is a popular distance between empirically estimated distributions, unlike the Kullback-Leibler divergence $D_{\text{KL}}(p \parallel q)$, which is often undefined. The JSD is defined as

$$D_{\text{JS}}(p \parallel q) := \frac{1}{2}\left(D_{\text{KL}}\left(p \parallel m\right) + D_{\text{KL}}\left(q \parallel m\right)\right)$$

where $m := \frac{1}{2}(p+q)$. The JSD measures the discriminability of two distributions: the average certainty with which a sample can be ascribed to one or the other.

To test the performance of a model p_i we compare the degree to which the histograms of the model samples match the corresponding histograms of the validation image, for all feature functions in the new set F_{i+1}. Let $\overline{\mathbf{h}_j(\mathbf{G})}$ denote the empirical probability distribution (normalised histogram) of the jth feature of F_{i+1} in an image $\mathbf{G}$. To score the overall goodness-of-fit $V(p_i|F_{i+1}) = \sum_{j=1}^{i+1} D_{\text{JS}}(\overline{\mathbf{h}_j(\mathbf{G}^{(i)})} \parallel \overline{\mathbf{h}_j(\mathbf{G}_{\text{valid}})})$ was used as the validation score function. This can be interpretered as estimating the total separability of the two groups of histograms when the feature functions are assumed to be independent from one another. Suppose $p^* = p_m$. Then V compares p_i and p^* across all the features shared by both as well as on the feature f^{m+1}, which was the one on which p^* was worst, and the feature f^{i+1} on which p_{i+1} is worst. This balances the pursuit of new statistics against the preservation of existing ones.

4.2 Sampling and learning parameters

Computing the expection in Eq. (3) exactly is intractable. Neither can it be reliably approximated using a Markov

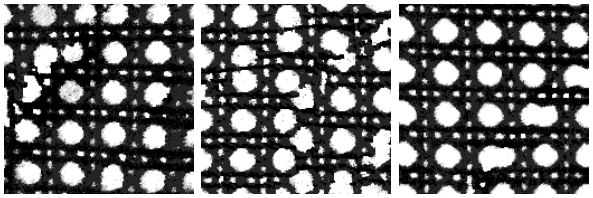

Figure 1: 170×170 samples from a model of D101 with 14 9th order BP features. *Left*: CSA without seed growing (inability to escape a local minimum; 'crystallisation'). *Centre*: seed growing but without CSA (the seed 'fractures' as it grows). *Right*: using CSA with seed growing usually but not always eliminates crystallisation, as in this example.

chain Monte Carlo (MCMC) sampler such as the Gibbs sampler, because the likelihood often has deep local minima. By accepting this and attempting only to draw approximate samples learning can be sped up. Controllable simulated annealling (CSA) [6] was used to simultaneously tune the parameters and produce approximate samples more quickly than possible using MCMC. CSA (later reinvented as persistent contrastive divergence (PCD) [20]) alternates Gibbs sampling macrosteps (updating each pixel in the image once) and changes to parameters; gradient; we used Robbins-Monro step sizes in our update rule $\Delta\boldsymbol{\theta} = 15(15 + t)^{-1}(\mathbf{f}(\mathbf{G}_{\text{samp}}) - \mathbf{f}(\mathbf{G}_{\text{obs}}))$, where $\mathbf{G}_{\text{samp}}$ is the current sample and t the iteration number. The same procedure has been used both for generating images (moving towards an image with $\mathbf{f}(\mathbf{G}_{\text{samp}}) \approx \mathbf{f}(\mathbf{G}_{\text{obs}})$), and for tuning parameters [20]. When CSA is used for image generation the final parameters are far from the MLE ones; however our experiments showed that under CSA parameters would often improve (move towards the MLE) for a while and then diverge again. To take advantage of this CSA was repeated three times for 100 iterations to learn parameters, each time restarting from an initial noise image. The three final images at $t = 100$ were used as the model samples in the nesting procedure.

Gibbs sampling from models of regular textures tends to get stuck in local minima due to 'crystallisation' of patterns growing and meeting without aligning (see Figure 1). To combat this we created a converged 'seed': by 15 macrosteps of Gibbs sampling on a 60×60 noise image, which was then grown on each edge by one pixel per iteration while running CSA. Using CSA during the growing phase is needed in order to make immediate corrections to halt the 'growing of garbage' outward. (Figure 1 centre). It is also necessary to trim the edges of the samples, where artifacts often appear due to being less constrained. A distance equal to the maximum size of any clique was trimmed from each sample, to obtains samples of size 110×110.

In order to synthesise texture images after modelling we used 350 iterations of an improved CSA using an adaptive step size as described by Almeida et al. [1] (with default hyperparameters and initial step size 1) instead of Robbins-Monro steps. All parameters in the model were free. This gave far faster convergence than Robbins-Monro steps, dampened oscillations, and is less sensitive to initial step size.

4.3 Binary patterns and feature selection

The high order features investigated were BPs with learnt offsets from the central pixel. No interpolation between pixels was performed, nor were histogram bins combined as in uniform LBPs [15]. The binary pattern feature function thresholds the grey levels of the pixels in each clique against a certain distinguished pixel (no longer necessarily in the centre): $f_{\text{BP}}(x_0, \ldots, x_d) = \sum_{1 \le i \le d} 2^{i-1} [\![x_0 < x_i]\!]$.

We write GLC^k and BP^k to indicate order k grey level co-occurrence and binary pattern features, respectively. All features were constrained to clique families with at most 40 pixels distance between two points. As base model, we used a MGRF with a first order factor to model marginal statistics and the two nearest neighbour GLC features with offsets $((1,0))$ and $((0,1))$. The initial candidate set C^1 was always comprised of all GLC^2 features with length ≤ 40. Thereafter different possibilities for C^2 were considered:

- GLC^3 features with cliques created by combining two of the selected GLC^2 feature cliques end-to-end (in any of the four possibilities).
- BP^5 features built out of the set of characteristic offsets $\{r_i\}$ occurring in the previously selected GLC^2 and BP^5 features. Each possible selection of two offsets r_1, r_2, and for each $i = 1, 2$ each choice of either using mirrored offsets $r_i, -r_i$ or halved offsets $r_i/2, -r_i/2$ provided the set of four-offset candidates.
- 'Simple' BP^9 features built by selecting four BP^3 features with symmetric pairs of offsets $(r_i, -r_i)$ of maximal training/sample JSD and then combining them into a single clique shape $(r_1, -r_1, \ldots, r_4, -r_4)$.
- 'Star' BP^9 features with 8 surrounding pixels evenly spaced on a circle centred on the x_0 pixel. Considing all rotations and radii from 1 to 20 pixels provided a fixed set of 144 candidates.

GLC^2 and BP^5 features were added two at a time for speedup.

5. SYNTHESIS EXPERIMENTS

25 greyscale textures from the Brodatz album [2] were selected which were diverse, difficult to model[1], homogeneous and without periodicity or other features on a scale longer than 40 pixels (the images were not scaled). These were subjectively categorised into three classes: stochastic (apparently described by only local interactions e.g. sand, water), near-regular (tiled but with random defects, e.g. weaves), and irregular (containing large scale elements with unpredictable placement or shapes e.g. marbles), as follows:

Stochastic	D4, D7, D9, D28, D32, D33, D37, D76, D84
Near-regular	D1, D14, D16, D20, D34, D52, D55, D56, D65, D82, D85, D101
Irregular	D22, D66, D68, D103

Each image was preprocessed with contrast-limited adaptive histogram equalization [26] to lessen the contrast-invariance advantage of the BP features over the GLC features and focus on structure learning, and quantised to $Q = 8$ grey

[1]See e.g. results of [16] at `http://www.cns.nyu.edu/~lcv/texture/`

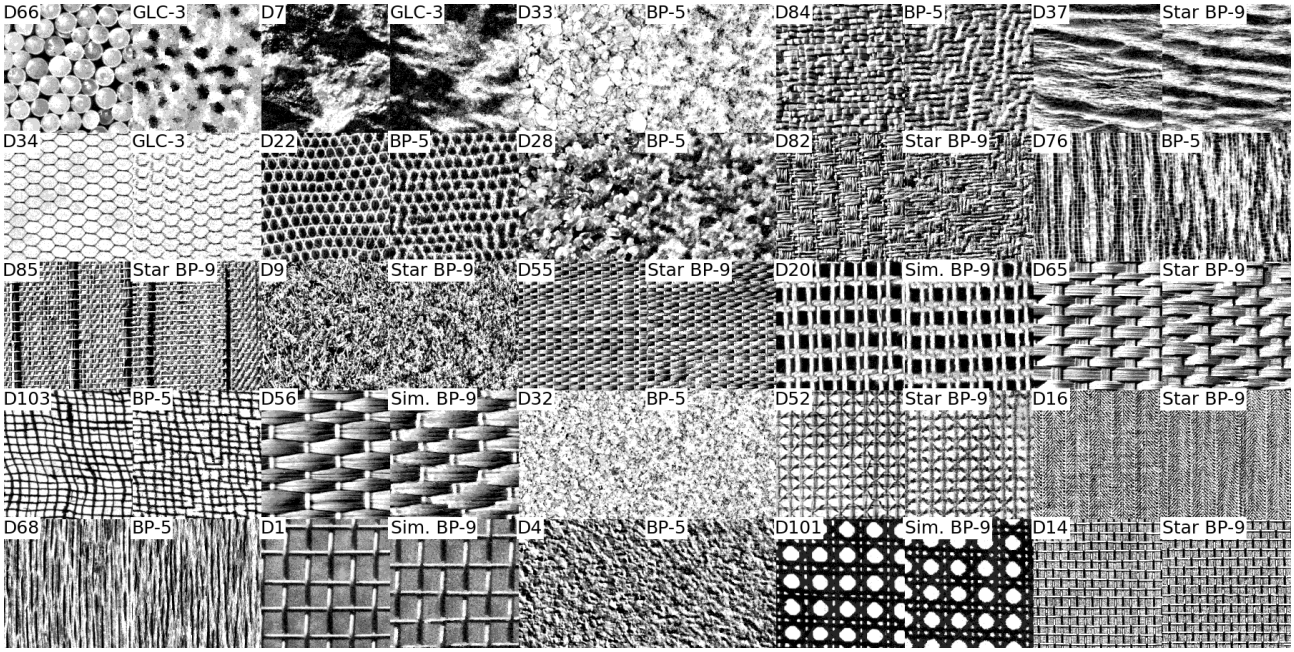

Figure 2: Training image and best synthesised sample for each of 25 Brodatz textures, according to average human evaluation score. Arranged in rows from worst average score (top left) to best (bottom right). The synthesised images are labelled with the scheme used to select their highest order features.

levels. Two adjacent 180×180 pieces were cut from each source image for training and validation. For each combination of selector described in Section 4.3 and texture a nested model was learnt three times and a 180×180 sample synthesised from each. In many cases no additional features were selected past the GLC^2 features in which case the model was excluded from synthesis and evaluation. Running times (a single threaded implementation) varied from 100s to 780s, typically averaging about 5 minutes, and the number of selection iterations varied from 10 to 26.

The samples from each model were scored by 17 people on a scale from 0 for no resemblance to 9 for indistinguishable. The observers were presented with the training image and up to 5 synthesised images for that texture at a time in a random order and instructed to make rapid decisions; each of the 216 synthesised images was presented once. There was huge variation between the scores assigned by different people; several images recieved both a 0 and a 9. The best synthesis result for each texture (out of the 4 to 15 samples for each texture shown to the observers) is shown in Figure 2. It can be seen that no feature selection scheme dominates the others, however models with 9th order potentials were best for 13 textures. Table 1 gives a summary of scores by texture category. When only some runs produced a model for a texture that went beyond GLC^2 features the reported mean and standard deviations are weighted averages with the scores obtained by the GLC^2 models. An all-pairs Nemenyi test indicated with over 99% certainty that BP^5 and star BP^9 models outperformed pairwise ones, and with over 95% certainty star BP^9 outperformed GLC^3 models.

We applied our nested MGRF model learning procedure to the task of synthesis of relatively high resolution textures

Table 1: Average human evaluations of texture synthesis results by texture category (*mean*±*standard deviation*).

	Stochas.	Near-reg.	Irregular	Total
GLC^2	3.7±1.8	3.6±1.9	3.1±1.5	3.5±1.8
GLC^3	4.2±1.9	4.0±1.9	3.6±1.6	4.0±1.9
BP^5	**4.7±1.9**	4.5±2.0	4.0±1.9	4.5±2.0
Sim. BP^9	3.8±1.9	4.8±2.1	3.5±1.7	4.3±2.0
Star BP^9	4.0±1.9	**5.6±2.0**	**4.4±1.8**	**4.8±1.9**

with texton and tessellation scales of up to about 40 pixels. This is in contrast to all of the recent filter-based MRF texture models [7, 9, 12, 17], which have only been demonstrated on simple highly regular textures with repetition and texton scales of up to about 15 pixels, using filters from only 7×7 to 11×11 in size, with the exception of successful synthesis of D16 in [12] by using multiple layers. These filter based models have also made use of GPGPU programming to speed up learning, while our learning procedure is practical on a single CPU core. Unfortunately the lack of difficult synthesis examples in these previously published approaches hinders comparison, though [12] achieved visually better results on the difficult D76 and D103 textures by using three layers of hundreds of filters. In contrast the best D76 and D103 models in Figure 2 both had 19 features.

6. DISCUSSION

The synthesis results show that the higher order models outperformed the pairwise models in capturing details. Interestingly the 5th order BP features were strong enough to perform well on stochastic textures, where the lack of structure and hence large amount of noise made the nesting algorithm

reject most higher order features. Surprisingly, those models built using star shaped LBP-like patterns (but with selected offset lengths and rotations) outperformed other forms of binary patterns selected in more sophisticated ways from much larger sets of candidates, including several others not reported in this paper for lack of space. However for many textures, especially stochastic ones, usually only pairwise features were selected, even if resulting models poorly modelled the texture. As a rule, the more features that were selected, the better the synthesis results, but the number selected were highly variable over repetitions. Clearly the details such as the stopping rule, can be improved upon. An iteration which by chance produces good CSA samples will set a bar too high for following iterations to exceed, causing the validation rule to stop the procedure.

Validation scoring of models specific to the desired application of the final model, rather than examining only its match to the sufficient statistics, would also be an interesting research direction.The statistical validation scoring that we used resulted in many features which yielded better visual similarity being rejected. For texture synthesis using a texture similarity function which closely matches human evaluation could be explored in future work. Several have recently been developed and validated for this purpose including Improved Structural Texture Similarity Metric (STSIM2) and a similarity metric based on Local Radial Index (LRI) [21, 24].

7. REFERENCES

[1] L. B. Almeida, T. Langlois, J. D. Amaral, and A. Plakhov. On-line learning in neural networks. chapter Parameter Adaptation in Stochastic Optimization, pages 111–134. Cambridge University Press, New York, 1998.

[2] P. Brodatz. *Textures: a Photographic Album for Artists and Designers*. Dover, New York, 1966.

[3] G. R. Cross and A. K. Jain. Markov random field texture models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5(1):25–39, Jan 1983.

[4] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[5] A. Gagalowicz and S. D. Ma. Sequential synthesis of natural textures. *Computer Vision, Graphics, and Image Processing*, 30(3):289 – 315, 1985.

[6] G. Gimel'farb. *Image Textures and Gibbs Random Fields*. Kluwer Academic Publishers, Dordrecht, 1999.

[7] N. Heess, C. K. I. Williams, and G. E. Hinton. Learning generative texture models with extended Fields-of-Experts. In *Proc. British Machine Vision Conf. (BMVC'09)*, 2009.

[8] E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106:620–630, May 1957.

[9] J. J. Kivinen and C. Williams. Multiple texture Boltzmann machines. In N. D. Lawrence and M. A. Girolami, editors, *Proc. 15th Int. Conf. on Artificial Intelligence and Statistics*, pages 638–646 vol.2, 2012.

[10] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. on Graphics, SIGGRAPH 2003*, 22(3):277–286, July 2003.

[11] S. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1-regularization. In *Advances in Neural Information Processing Systems 19 (NIPS'06)*, 2006.

[12] H. Luo, P. L. Carrier, A. Courville, and Y. Bengio. Texture modeling with convolutional spike-and-slab RBMs and deep extensions. *J. of Machine Learning Research*, 31:415–423, 2013.

[13] L. Mihalkova and R. J. Mooney. Bottom-up learning of Markov logic network structure. In *Proc. 24th Int. Conf. on Machine Learning*, ICML '07, pages 625–632, New York, 2007. ACM.

[14] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996.

[15] T. Ojala, M. Pietikäinen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7):971–987, Jul 2002.

[16] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Computer Vision*, 40(1):49–70, 2000.

[17] M. Ranzato, V. Mnih, and G. E. Hinton. Generating more realistic images using gated MRFs. In *Advances in Neural Information Processing Systems 23 (NIPS'10)*, pages 2002–2010. Curran Associates, 2010.

[18] S. Roth and M. J. Black. Fields of Experts. *Int. J. of Computer Vision*, 82(2):205–229, 2009.

[19] M. Schmidt and K. Murphy. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proc. Int. Workshop on Artificial Intelligence and Statistics*, 2010.

[20] T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proc. 25th Int. Conf. on Machine Learning (ICML'08)*, pages 1064–1071. ACM, 2008.

[21] J. Žujović. *Perceptual Texture Similarity Metrics*. PhD thesis, Northwestern University, 2011.

[22] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proc. 27th Conf. on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 479–488, New York, 2000. ACM Press/Addison-Wesley Publishing.

[23] A. Zalesny. Analysis and synthesis of textures with pairwise signal interactions. Technical Report KUL/ESAT/PSI/9902, Katholieke Universiteit Leuven, 1999.

[24] Y. Zhai, D. Neuhoff, and T. Pappas. Local radius index - a new texture similarity feature. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1434–1438, May 2013.

[25] S. C. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *Int. J. of Computer Vision*, 27(2):107–126, 1998.

[26] K. Zuiderveld. Contrast limited adaptive histogram equalization. In *Graphic Gems IV*, pages 474–485. Academic Press Professional, 1994.