Lecture 13: Images as Compositions of Patterns

A.L. Yuille

March 13, 2012

1 Introduction

Pattern theory is a research program which believes that visual processing is best thought of in terms of patterns (Grenander 1973, Mumford and Desolneux 2010). Images can be expressed as combinations of elementary patterns, see figure (1), and image interpretation/understanding consists of decomposing the image into these patterns. The same principles apply to other sensory systems and to cognition and thought. This program is very attractive but it is also very difficult. Here we restrict ourself to applying it to image segmentation with a small extension to image parsing which includes the detection and recognition of objects. Note that the weak membrane/smooth models of images (earlier lectures) can be thought of as examples of this approach but where the patterns are very simple – image regions with smooth intensity. The pattern theory approach suggests that we have a larger dictionary of patterns – including texture, shading, and even patterns for objects such as faces and text. See also Zhu and Mumford (2006).

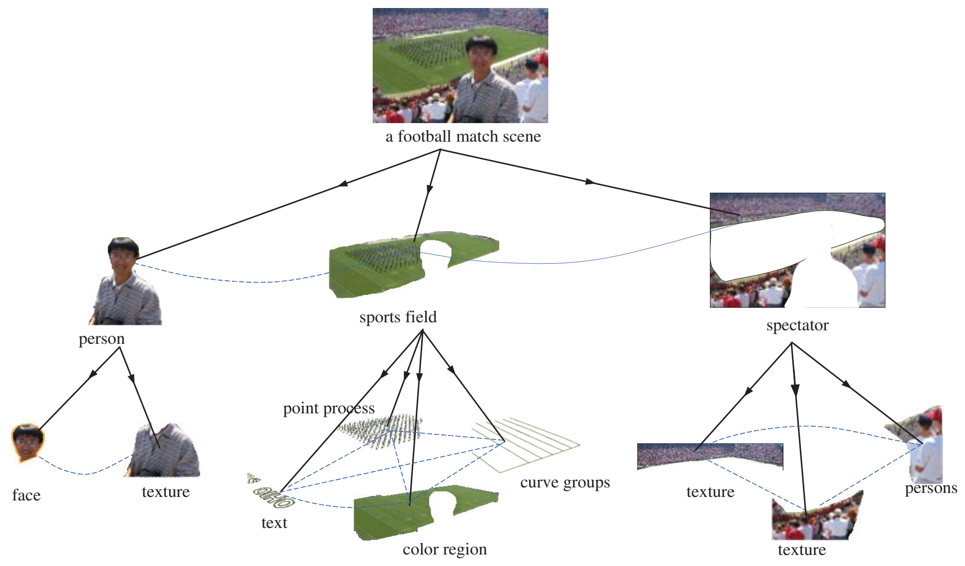


Figure 1: Pattern theory argues that images are composed from different patterns and the task of image processing is to parse the image into its constituent patterns.

This lecture will start by describing snakes (Witkin, Terzopoulos, Kass) which relate to this framework but pre-date it. This is one of the most cited papers in computer vision. We will formulate it probabilistically, which relates to a model for detecting roads by Geman and Jedynak.

After describing snakes, we will proceed by describing three pattern theory models of images of increasing complexity: (i) Zhu and Yuille (1996), (ii) Tu and Zhu (2002), and (iii) Tu, Chen, Yuille, Zhu (2005). The last two require complex algorithms – data driven Markov Chain Monte Carlo (DDMCMC), which we will also describe. We will introduce (ii) and (iii) together since the mathematics is very similar.

2 Snakes

Snakes were originally intended as an interactive tool. It has many applications and, in particular, it motivated considerable work on object tracking (see "Active Vision" A. Blake and A.L. Yuille eds.).

2.1 Basic Snakes

A snake is a contour by $\Gamma(s)$ (s is not the arc length). Typically $\Gamma(s)$ is a closed contour. It is defined by an energy function:

$$E[\Gamma] = \int_\Gamma \frac{1}{2}\{\alpha|\vec{\Gamma}_s(s)|^2 + \beta|\vec{\Gamma}_{ss}|^2 - \lambda|\vec{\nabla} I(\vec{x}(s))|^2\}ds. \tag{1}$$

Here $_s$ denotes partial derivative with respect to s – $\Gamma_s = \frac{\partial\vec{\Gamma}}{\partial s}$.

To run a snake, you initialize it by placing it near, or surrounding, and object (e.g. a face). Then minimize $E[\Gamma]$ by steepest descent. This gives update equations:

$$\frac{d\vec{\Gamma}(s)}{dt} = -\alpha\vec{\Gamma}_{ss} + \beta\vec{\Gamma}_{ssss} + \lambda\vec{\nabla}|\vec{\nabla} I|^2. \tag{2}$$

The first two terms try to make the curve short and have small curvature (NEED AN APPENDIX ON CURVATURE.). The third term tries to move the snake to regions of large intensity gradient.

Snakes can be modified in several different ways. Basic snakes try to minimize their areas (consequence of the prior terms). But we can have *balloons* by putting in prior terms that try to make the area as big as possible. We can also add regional terms (check Ishihara and Geiger!!). Alternative algorithms are more effective than steepest descent.

2.2 Bayesian Interpretation of Snakes

Bayesian interpretation. There is a natural Bayesian interpretation of snakes. This shows that the energy imaging term $-|\vec{\nabla} I|^2$ is not very sensible (as was realized in practice fairly soon). Technically this requires taking the continuous formulation and discretizing it.

Specify a generative model for the image intensity gradients (see notes for lecture 1):

$$P(\{|\vec{\nabla} I|\}|\Gamma) = \prod_{\vec{x}\in\Gamma} P_{on}(|\vec{\nabla} I(\vec{x})|) \prod_{\vec{x}\in\Omega/\Gamma} P_{off}(|\vec{\nabla} I(\vec{x})|). \tag{3}$$

$$P(\Gamma) = \frac{1}{Z}\exp\{-\int_\Gamma \frac{1}{2}\{\alpha|\vec{\Gamma}_s(s)|^2 + \beta|\vec{\Gamma}_{ss}|^2\}\}. \tag{4}$$

MAP estimation of Γ corresponds to minimizing:

$$-\log P(\{|\vec{\nabla} I|\}|\Gamma) - \log P(\Gamma), \tag{5}$$

where we see that the second term reduces to the spatial terms in the energy function for the snakes.

We now concentrate on the first term which can be re-expressed as:

$$-\sum_{\vec{x}\in\Gamma} \log P_{on}(|\vec{\nabla} I(\vec{x})|) - \sum_{\vec{x}\in\Omega/\Gamma} \log P_{off}(|\vec{\nabla} I(\vec{x})|) = -\sum_{\vec{x}\in\Gamma} \log \frac{P_{on}(|\vec{\nabla} I(\vec{x})|)}{P_{off}(|\vec{\nabla} I(\vec{x})|)} - \sum_{\vec{x}\in\Omega} P_{off}(|\vec{\nabla} I(\vec{x})|). \tag{6}$$

We see that the second term is independent of Γ and hence can be dropped during MAP estimation. We then see that $\log \frac{P_{on}(|\vec{\nabla} I(\vec{x})|)}{P_{off}(|\vec{\nabla} I(\vec{x})|)}$ corresponds to the term $-|\vec{\nabla} I(\vec{x})|^2$. But we know what the typical shape of the log-likelihood function is from lecture 1 – and it certainly is not $-|\vec{\nabla} I(\vec{x})|^2$! Instead it rises slowly from a minimum at $|\vec{\nabla} I| = 0$ and (roughly) asymptotes. This is because if $|\vec{\nabla} I|$ is large then it is almost certainly an edge – but $-|\vec{\nabla} I(\vec{x})|^2$ rewards big edges too much at the expense of small edges.

Models like equation (6) were used, in combination with a smoothness prior, to model roads in aerial images (Geman and Jedynak).

3 Probabilistic Models of Images: (I) Region Competition

The goal of image segmentation is to decompose the image domain Ω into M non-overlapping sub-domains $\{\Omega_i : i = 1, ..., M\}$ – such that $\bigcup_{i=1}^{M} \Omega_i = \Omega$ and $\Omega_i \bigcap \Omega_j = 0,\ i \neq j$. The boundaries of sub-region i is specified by $\partial \Omega_i$. We defined Γ to be the set of all the boundaries. The number of regions M is a random variables – we do not know how many regions there will be in any image.

The region competition model (Zhu and Yuille 1996) assumes that the intensity, or intensity features, in each subregion Ω_i are generated by a distribution parameterized by α_i. Formally we write $P(\{I(x,y) : (x,y) \in \Omega_i\}|\alpha_i)P(\alpha_i)$. For example, a simple model is that the image intensities are drawn independently at each position (x,y) from a Gaussian distribution – i.e. $\alpha_i = (\mu_i, \sigma_i^2)$, $P(\{I(x,y) : (x,y) \in \Omega_i\}|\mu_i, \sigma_i) = \prod_{(x,y)\in\Omega_i} N(I(x,y)|\mu_i, \sigma_i^2)$, where $N(I(x,y)|\mu_i, \sigma_i^2)$ is a Gaussian with mean μ_i and variance σ_i^2. By extending this formulation, we can also include the imaging term for the Mumford-Shah model – α_i corresponds to $\{J(x,y) : (x,y) \in \Omega_i\}$ and there is a Gaussian prior defined on J specified by $(1/Z)\exp\{-\int\int_{\Omega_i} dxdy \vec{\nabla} J(x,y) \cdot \vec{\nabla} J(x,y)\}$. (Note: Zhu's work on minimax entropy learning was driven by the need to discover other, more realistic, models of images that could be used).

Putting everything together gives a model:

$$P(\{\Omega_i\}, M, \{\alpha_i\}) = \frac{1}{Z}\exp\{-E[\{\Omega_i\}, \{\alpha_i\}, M]\}, \tag{7}$$

where

$$E[\{\Omega_i\}, \{\alpha_i\}, M] = \sum_{i=1}^{M} \frac{\mu}{2}\int_{\partial\Omega_i} ds_{\partial\Omega_i} + \lambda M - \sum_{i=1}^{M}\log P(\{I(x,y) : (x,y) \in \Omega_i\}|\alpha_i) - \sum_{i=1}^{M}\log P(\alpha_i). \tag{8}$$

We can express the model in a, more insightful, generative form:

$$P(I|\Omega, \alpha)P(\alpha)P(\Omega|M)P(M), \tag{9}$$

where

$$P(M) = \frac{1}{Z_1}\exp\{-\lambda M\},$$
$$P(\Omega|M) = \frac{1}{Z_2}\exp\{-(\mu/2)\sum_{i=1}^{M}\int_{\partial\Omega_i} ds\},$$
$$P(I|\Omega, \alpha) = \prod_{i=1}^{N} P(\{I(x,y) : (x,y) \in \Omega_i\}|\alpha_i). \tag{10}$$

In other words, to generate an image you proceed in four steps: (I) sample M from $P(M)$ – to generate the number of regions in the image, (II) sample the shape of the regions $\{\Omega_i\}$ from $P(\Omega|M)$, (III) sample the parameters α_i for each region from $P(\alpha)$, and (IV) sample the images in each region from $P(\{I(x,y) : (x,y) \in \Omega_i\}|\alpha_i)$.

In practice, it is very hard to do this sampling. The most difficult step is to sample from $P(\Omega|M)$ to generate a partition of the image into regions (there have been few attempts to do this). But, using the more advanced model in the next section, Tu and Zhu (2002) did from $P(\{I(x,y) : (x,y) \in \Omega_i\}|\alpha_i)$ when the α_i's were estimated from the images (their sampled results often looked similar to the input images).

We can also interpret this model in terms of encoding the image. Shannon's information theory proposes that if data is generated by a distribution $P(x)$ then an example x should be encoded by $-\log P(x)$ bits. If the distribution has a parameter α, then it is best encoded by finding $\alpha^* = \arg\min_\alpha\{-\log P(x|\alpha)\}$, which is simply the maximum likelihood estimate. (recall discussion of information theory in earlier lectures).

3.1 Inference: Region Competition

The region competition algorithm was proposed to minimize the energy function in equation (8). The algorithm is not guaranteed to converge to the global optimal solution because the energy function is highly non convex.

Suppose the number M of regions is known. Then we can fix M and minimize $E(\Omega, \alpha, M)$ with respect to Ω and α alternatively.

This gives two (alternating) steps:

$$\text{Solve } \alpha_i^* = \arg\max\{\int\int_{\Omega_i} \log P(\alpha_i|I(x,y))dxdy\}, \quad \forall i = 1, ..., M. \tag{11}$$

$$\frac{d\Gamma(s)}{dt} = \frac{-\mu}{2}\kappa_\nu \vec{n}_\nu + \log\frac{P(I(\nu)|\alpha_k)}{P(I(\nu)|\alpha_{k+1})}\vec{n}_\nu. \tag{12}$$

Here ν denotes a point of the boundary Γ between regions k and $k+1$, $\vec{n}_\nu$ is the normal to the boundary curve. This equation is derived as gradient descent of the functional (calculus of variations).

The intuitions for these equations are clear. The regions k and $k+1$ compete for "ownership" of the pixels on the boundary between the regions (by the log-likelihood term) and the curvature terms tries to make the boundary as short (straight) as possible. In particular, if the probability distributions are Gaussians we obtain:

$$\frac{d\Gamma(s)}{dt} = \frac{-\mu}{2}\kappa_\nu \vec{n}_\nu - \frac{1}{2}\{\log\frac{\sigma_i^2}{\sigma_j^2} + \frac{(\overline{I} - \mu_i)^2}{\sigma_i^2} - \frac{(\overline{I} - \mu_j)^2}{\sigma_j^2} + \frac{s^2}{\sigma_i^2} - \frac{s^2}{\sigma_j^2}\}\vec{n}_\nu, \tag{13}$$

where $\overline{I}$ is the intensity mean in the window W and s^2 is the intensity variance.

Note: In practice, a window is used to get better estimate of the statistics (Zhu et al) – so we replace the log-likelihood term on the pixel by the log-likelihood over the window. Other example involves texture statistics I_x, I_y.

But how to estimate the number M of regions? The following strategy was used (Zhu et al 1996): (i) initialize with many (small) regions, (ii) run the 2-stage iterative algorithm – this will make some regions disappear (get squashed) while others grow large, (iii) remove the squashed regions, (iv) merge bigger regions if their α's are similar and if this decreases the energy (tradeoffs – need to estimate a single α for the combined region, which increases the energy, but this may be compensated for by eliminating the boundary between the regions and the reduced penalty for number of regions).

4 Probabilistic Models of Images: (II) Image Parsing

We now introduce a richer class of image models. The energy function in equation (8) was limited because it used the same image model for each image region. The next step (Tu and Zhu 2002) was to allow there to be several different models which could be used for each image. Formally, this required having families of models – $P(I \in \Omega_i|\tau, \alpha)P(\alpha|\tau)P(\tau)$, where τ is the type of the model and α are its parameters. This allows us to use models for texture, for smoothly varying intensity, for junk, or for image regions containing smooth gradients. This model represents the images better but it leads to a more complex inference problem – we know have to estimate the model class τ as well as the model parameters α. Note that there the image models can be extended to include models of objects, as was done in (Tu, Chen, Yuille, Zhu 2005) These generative models are illustrated in figure (2)(left panel). The effects of image parsing are shown in figure (**??**).

The DDMCMC algorithm was proposed to perform inference for this richer class of models (Tu and Zhu 2002). Formally, the DDMCMC algorithm simulates a Markov chain $\mathcal{MC} =< \Omega, \nu, \mathcal{K} >$ with kernel $\mathcal{K}$ in space Ω and with probability ν for the starting state. An element $W \in \Omega$ is a parsing graph. We proceed by defining a set of moves for reconfiguring the graph. These include moves to: (i) create nodes, (ii) delete nodes, and (iii) change node attributes. We specify stochastic dynamics for these moves in terms of transition kernels. Some types of moves are illustrated in figure (2)(right panel).

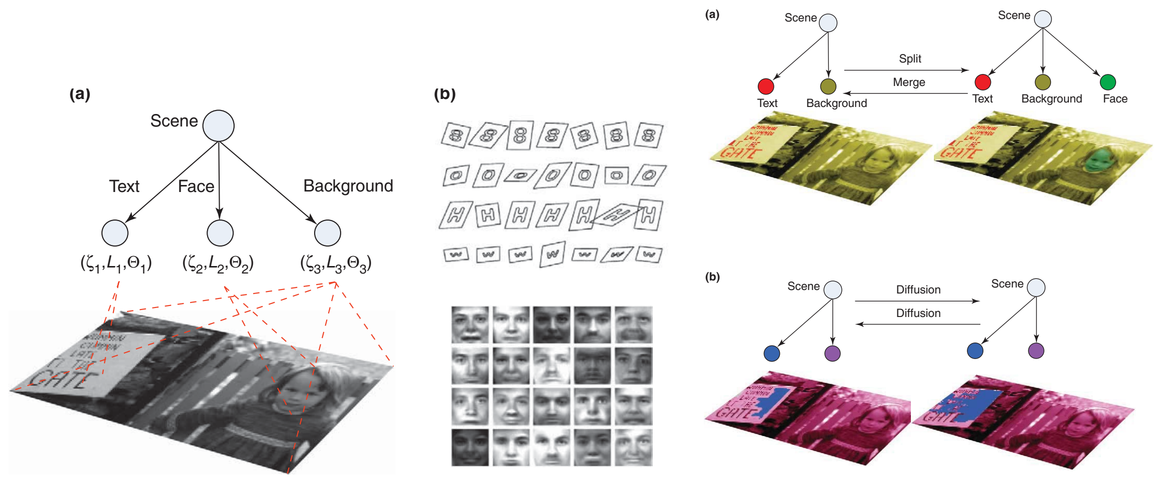


Figure 2: Left Panel: A generative model for images (a) and samples from models of objects – letters and faces (b). Right panel: examples of moves which allow us to search through the space of W.

For each move we define a Markov Chain sub-kernel by a transition matrix $\mathcal{K}_a(W'|W : \mathbf{I})$ with $a \in \mathcal{A}$ being an index.

Each sub-kernel[1] is designed to be of Metropolis-Hastings form:

$$\mathcal{K}_a(W'|W : \mathbf{I}) = Q_a(W'|W : \mathrm{Tst}_a(\mathbf{I})) \min\{1, \frac{p(W'|\mathbf{I})Q_a(W|W' : \mathrm{Tst}_a(\mathbf{I}))}{p(W|\mathbf{I})Q_a(W'|W : \mathrm{Tst}_a(\mathbf{I}))}\}, \quad W' \neq W \tag{14}$$

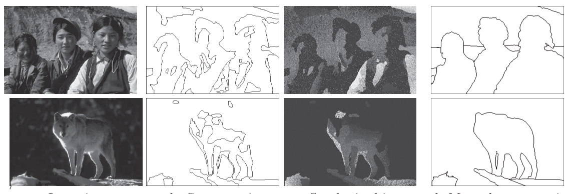
where a transition from W to W' is proposed (stochastically) by the proposal probability $Q_a(W'|W : \mathrm{Tst}_a(\mathbf{I}))$ which depends on features/tests $\mathrm{Tst}_a(\mathbf{I}))$ performed on the image. These tests are accepted (stochastically) by the acceptance probability:

$$\alpha(W'|W : \mathbf{I}) = \min\{1, \frac{p(W'|\mathbf{I})Q_a(W|W' : \mathrm{Tst}_a(\mathbf{I}))}{p(W|\mathbf{I})Q_a(W'|W : \mathrm{Tst}_a(\mathbf{I}))}\}. \tag{15}$$

For more details about MCMC See the appendix.

The effectiveness of DDMCMC (i.e. its convergence rate) depends on the choice of the proposal probabilities. In fact this is true of many real world applications of MCMC. So there is a separate modeling stage which involves finding good proposals. This involves finding cues for the probable positions of edges in the image, for the detection of faces and text.

[1]Except for one that evolves region boundaries.



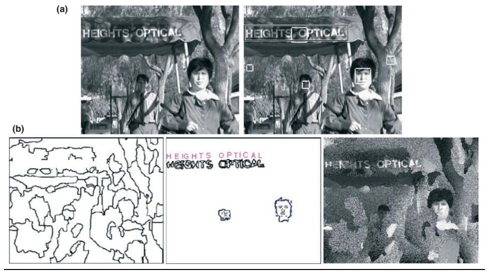


Figure 3: Left Panel: the need for object models to obtain good segmentation. The input image (a), segmentation by a model (Tu znd Zhu) which does not use models of objects, (c) the image re-sampled from the Tu and Zhu model, (d) the groundtruth segmentation. Right Panel: the effect of using object models. The input image (upper left) and the proposals for faces and text (upper right). The segmentation (lower left), the text and faces detected (lower center), and the re-sampled image (lower right).

4.1 Generative Models

The number K of intermediate nodes is a random variable, and each node $i = 1, ..., K$ has a set of attributes (L_i, ζ_i, Θ_i) defined as follows. L_i is the shape descriptor and determines the region $R_i = R(L_i)$ of the image pixels covered by the visual pattern of the intermediate node. Conceptually, the pixels within R_i are child nodes of the intermediate node i. (Regions may contain holes, in which case the shape descriptor will have internal and external boundaries). The remaining attribute variables (ζ_i, Θ_i) specify the probability models $p(\mathbf{I}_{R(L_i)}|\zeta_i, L_i, \Theta_i)$ for generating the sub-image $\mathbf{I}_{R(L_i)}$ in region $R(L_i)$. The variables $\zeta_i \in \{1, ..., 66\}$ indicate the visual pattern type (3 types of generic visual patterns, 1 face pattern, and 62 text character patterns), and Θ_i denotes the model parameters for the corresponding visual pattern (details are given in the following subsections). The complete scene description can be summarized by:

$$W = (K, \{(\zeta_i, L_i, \Theta_i) : i = 1, 2, ..., K\}).$$

The shape descriptors $\{L_i : i = 1, ..., K\}$ are required to be consistent so that each pixel in the image is a child of one, and only one, of the intermediate nodes. The shape descriptors must provide a partition of the image lattice $\Lambda = \{(m, n) :\ 1 \leq m \leq Height(\mathbf{I}), 1 \leq n \leq Width(\mathbf{I})\}$ and hence satisfy the condition

$$\Lambda = \cup_{i=1}^{K} R(L_i), \quad R(L_i) \cap R(L_j) = \emptyset, \quad \forall i \neq j.$$

The generation process from the scene description W to $\mathbf{I}$ is governed by the likelihood function:

$$p(\mathbf{I}|W) = \prod_{i=1}^{K} p(\mathbf{I}_{R(L_i)}|\zeta_i, L_i, \Theta_i).$$

The prior probability $p(W)$ is defined by

$$p(W) = p(K) \prod_{i=1}^{K} p(L_i) p(\zeta_i|L_i) p(\Theta_i|\zeta_i).$$

In our Bayesian formulation, parsing the image corresponds to computing the W^* that maximizes *a posteriori* probability over Ω, the solution space of W,

$$W^* = \arg\max_{W \in \Omega} p(W|\mathbf{I}) = \arg\max_{W \in \Omega} p(\mathbf{I}|W) p(W). \tag{16}$$

It remains to specify the prior $p(W)$ and the likelihood function $p(\mathbf{I}|W)$. We set the prior terms $p(K)$ and $p(\Theta_i|\zeta_i)$ to be uniform probabilities. The term $p(\zeta_i|L_i)$ is used to penalize high model complexity.

4.2 Generative intensity models

We use four families of generative intensity models for describing intensity patterns of (approximately) constant intensity, clutter/texture, shading, and faces (a fifth model for letters is omitted for reasons of space). The first three are similar to those defined in (Tu and Zhu 2002).

1. Constant intensity model $\zeta = 1$:.

This assumes that pixel intensities in a region R are subject to independently and identically distributed (iid) Gaussian distribution,

$$p_1(\mathbf{I}_{R(L)}|\zeta = 1, L, \Theta) = \prod_{v \in R(L)} G(\mathbf{I}_v - \mu; \sigma^2), \quad \Theta = (\mu, \sigma)$$

2. Clutter/texture model $\zeta = 2$:.

This is a non-parametric intensity histogram $h()$ discretized to take G values (i.e. is expressed as a vector $(h_1, h_2, ..., h_G)$). Let n_j be the number of pixels in $R(L)$ with intensity value j.

$$p_2(\mathbf{I}_{R(L)}|\zeta = 2, L, \Theta) = \prod_{v \in R(L)} h(\mathbf{I}_v) = \prod_{j=1}^{G} h_j^{n_j}, \quad \Theta = (h_1, h_2, ..., h_G).$$

3. Shading model $\zeta = 3$ and $\zeta = 5, ..., 66$:.

This family of models are used to describe generic shading patterns, and text characters. We use a quadratic form

$$J(x, y; \Theta) = ax^2 + bxy + cy^2 + dx + ey + f,$$

with parameters $\Theta = (a, b, c, d, e, f, \sigma)$. Therefore, the generative model for pixel (x, y) is

$$p_3(\mathbf{I}_R(L)|\zeta \in \{3, (5, ..., 66)\}, L, \Theta) = \prod_{v \in R(L)} G(\mathbf{I}_v - J_v; \sigma^2), \quad \Theta = (a, b, c, d, e, f, \sigma).$$

4. The PCA face model $\zeta = 4$:.

The generative model for faces is simpler and uses Principal Component Analysis (PCA) to obtain representations of the faces in terms of principal components $\{B_i\}$ and covariances Σ. Lower level features, also modeled by PCA, can be added. Figure **??** shows some faces sampled from the PCA model.

$$p_4(\mathbf{I}_R(L)|\zeta = 4, L, \Theta) = G(\mathbf{I}_{R(L)} - \sum_i \lambda_i B_i; \Sigma), \quad \Theta = (\lambda_1, .., \lambda_n, \Sigma).$$

5. Models of Text $\zeta = 5$:. Conceptually straightfroward. Too complex to include.

4.3 Proposal Probabilities

The proposal probabilities $q(w_j|\text{Tst}_j(\mathbf{I}))$ make proposals for the elementary components w_j of W. For computational efficiency, these probabilities are based only on a small number of simple tests $\text{Tst}_j(\mathbf{I})$. The cues are of the following forms:

1. Edge Cues. These cues are based on edge detectors. They are used to give proposals for region boundaries (i.e. the shape descriptor attributes of the nodes). Specifically, we run the Canny detector at three scales followed by edge linking to give partitions of the image lattice. This gives a finite list of candidate partitions which are assigned weights, see section (5.2.3). The probability is represented by this weighted list of particles (similar to particle filtering).

2. Binarization Cues. These cues are computed using a variant of Niblack's algorithm. They are used to propose boundaries for text characters (i.e. shape descriptors for text nodes), and will be used in conjunction with proposals for text detection. Like edge cues, the algorithm is run at different parameters settings and represents the discriminative probability by a weighted list of particles indicating candidate boundary locations.

3. Face Region Cues. These cues are learnt by a variant of AdaBoost which outputs probabilities (Hastie, Tibishani, Friedman). They propose the presence of faces in sub-regions of the image. These cues are combined with edge detection to propose the localization of faces in an image.

4. Text Region Cues. These cues are also learnt by a probabilistic version of AdaBoost. The algorithm is applied to image windows (at a range of scales). It outputs a discriminative probability for the presence of text in each window. Text region cues are combined with binarization to propose boundaries for text characters.

5. Shape Affinity Cues. These act on shape boundaries, produced by binarization, to propose text characters. They use shape context cues and information features to propose matches between the shape boundaries and the deformable template models of text characters.

6. Region Affinity Cues. These are used to estimate whether two regions R_i, R_j are likely to have been generated by the same visual pattern family and model parameters. They use an affinity similarity measure of the intensity properties $\mathbf{I}_{R_i}, \mathbf{I}_{R_j}$.

7. Model Parameter and Visual Pattern Family cues. These are used to propose model parameters and visual pattern family identity. They are based on clustering algorithms, such as mean-shift. .

In our current implementation, we conduct all the bottom-up tests $\mathrm{Tst}_j(\mathbf{I}), j = 1, 2, ..., K$ at an early stage for all the probability models $q_j(w_j|\mathrm{Tst}_j(\mathbf{I}))$, and they are then combined to form composite tests $\mathrm{Tst}_a(\mathbf{I})$ for each subkernel $\mathcal{K}_a$ in equations (14,15).

4.4 Control Structure of the Algorithm

The control strategy used by our image parser explores the space of parsing graphs by a Markov Chain Monte Carlo sampling algorithm. This algorithm uses a transition kernel $\mathcal{K}$ which is composed of sub-kernels $\mathcal{K}_a$ corresponding to different ways to reconfigure the parsing graph. These sub-kernels come in reversible pairs[2] (e.g. birth and death) and are designed so that the target probability distribution of the kernel is the generative posterior $p(W|\mathbf{I})$. At each time step, a sub-kernel is selected stochastically. The sub-kernels use the Metropolis-Hasting sampling algorithm, see equation (14), which proceeds in two stages. First, it proposes a reconfiguration of the graph by sampling from a proposal probability. Then it accepts or rejects this reconfiguration by sampling the acceptance probability.

To summarize, we outline the control strategy of the algorithm below. At each time step, it specifies (stochastically) which move to select (i.e. which sub-kernel), where to apply it in the graph, and whether to accept the move. The probability to select moves $\rho(a : \mathbf{I})$ was first set to be independent of $\mathbf{I}$, but we got better performance by adapting it using discriminative cues to estimate the number of faces and text characters in the image (see details below). The choice of where to apply the move is specified (stochastically) by the sub-kernel. For some sub-kernels it is selected randomly and for others is chosen based on a *fitness factor* (see details in section (5)), which measures how well the current model fits the image data.

[2]Except for the boundary evolution sub-kernel which will be described separately, see Section 5.1.

The basic control strategy of the image parsing algorithm

1. Initialize W (e.g. by dividing the image into four regions), setting their shape descriptors, and assigning the remaining node attributes at random.
2. Set the temperature to be T_{init}.
3. Select the type a of move by sampling from a probability $\rho(a)$, with $\rho(1) = 0.2$ for faces, $\rho(2) = 0.2$ for text, $\rho(3) = 0.4$ for splitting and merging, $\rho(4) = 0.15$ for switching region model (type or model parameters), and $\rho(5) = 0.05$ for boundary evolution. This was modified slightly adaptively, see caption and text.
4. If the selected move is boundary evolution, then select adjacent regions (nodes) at random and apply stochastic steepest descent, see section (5.1).
5. If the jump moves are selected, then a new solution W' is randomly sampled as follows:

   – For the birth or death of a face, see section (5.2.2), we propose to create or delete a face. This includes a proposal for where in the image to do this.

   – For the birth of death of text, see section (5.2.1), we propose to create a text character or delete an existing one. This includes a proposal for where to do this.

   – For region splitting, see section (5.2.3), a region (node) is randomly chosen biased by its fitness factor. There are proposals for where to split it and for the attributes of the resulting two nodes.

   – For region merging, see section (5.2.3), two neighboring regions (nodes) are selected based on a proposal probability. There are proposals for the attributes of the resulting node.

   – For switching, see section (5.2.4), a region is selected randomly according to its fitness factor and a new region type and/or model parameters is proposed.

   - The full proposal probabilities, $Q(W|W : \mathbf{I})$ and $Q(W'|W : \mathbf{I})$ are computed.
   - The Metropolis-Hastings algorithm, equation (14), is applied to accept or reject the proposed move.

6. Reduce the temperature $T = 1 + T_{init} \times exp(-t \times c|R|)$, where t is the current iteration step, c is a constant and $|R|$ is the size of the image.
7. Repeat the above steps and until the convergence criterion is satisfied (by reaching the maximum number of allowed steps or by lack of decrease of the negative log posterior).

5 The Markov Chain kernels

This section gives a detailed discussion of the individual Markov Chain kernel, their proposal probabilities, and their fitness factors.

5.1 Boundary Evolution

These moves evolve the positions of the region boundaries but preserve the graph structure. They are implemented by a stochastic partial differential equation (Langevin equation) driven by Brownian noise and can be derived from a Markov Chain. The deterministic component of the PDE is obtained by performing steepest descent on the negative log-posterior, as derived in (Zhu and Yuille 1996).

We illustrate the approach by deriving the deterministic component of the PDE for the evolution of the boundary between a letter T_j and a generic visual pattern region $\mathbf{R}_i$. The boundary will be expressed in terms of the control points $\{S_m\}$ of the shape descriptor of the letter. Let v denote a point on the boundary, i.e. $v(s) = (x(s), y(s))$ on $\Gamma(s) = \partial R_i \cap \partial R_j$. The deterministic part of the evolution equation is obtained by taking the derivative of the negative log-posterior $-\log p(W|\mathbf{I})$ with respect to the control points.

More precisely, the relevant parts of the negative log-posterior, see equation (16,4.1) are given by $E(\mathbf{R}_i)$ and $E(T_j)$ where:

$$E(\mathbf{R}_i) = \int \int_{R_i} \{-\log p(\mathbf{I}(x, y)|\theta_{\zeta_i})\} dxdy + \gamma |R_i|^{\alpha} + \lambda |\partial R_i|.$$

and

$$E(T_j) = \int\int_{L_j} \log p(\mathbf{I}(x,y)|\theta_{\zeta_j})dxdy + \gamma|R(L_j)|^{\alpha} - \log p(L_j).$$

Differentiating $E(R_i) + E(T_j)$ with respect to the control points $\{S_m\}$ yields the evolution PDE:

$$\begin{aligned}
\frac{dS_m}{dt} &= -\frac{\delta E(\mathbf{R}_i)}{\delta S_m} - \frac{\delta E(T_j)}{\delta S_m} \\
&= \int [-\frac{\delta E(\mathbf{R}_i)}{\delta v} - \frac{\delta E(T_j)}{\delta v}]\frac{1}{|\mathbf{J}(s)|}ds \\
&= \int \mathbf{n}(v)[\log \frac{p(\mathbf{I}(v);\theta_{\zeta_i})}{p(\mathbf{I}(v);\theta_{\zeta_j})} + \alpha\gamma(\frac{1}{|D_j|^{1-\alpha}} - \frac{1}{|D_i|^{1-\alpha}}) - \lambda\kappa + D(G_{S_j}(s)||G_T(s))]\frac{1}{|\mathbf{J}(s)|}ds,
\end{aligned}$$

where $\mathbf{J}(s)$ is the Jacobian matrix for the spline function. (Recall that $\alpha = 0.9$ in the implementation).

The log-likelihood ratio term $\log \frac{p(\mathbf{I}(v);\theta_{\zeta_i})}{p(\mathbf{I}(v);\theta_{\zeta_j})}$ implements the competition between the letter and the generic region models for ownership of the boundary pixels.

5.2 Markov chain sub-kernels

Changes in the graph structure are realized by Markov chain *jumps* implemented by four different sub-kernels.

5.2.1 Sub-kernel I: birth and death of text

This pair of jumps is used to create or delete text characters. We start with a parse graph W and transition into parse graph W' by creating a character. Conversely, we transition from W' back to W by deleting a character.

The proposals for creating and deleting text characters are designed to approximate the terms in equation (**??**). We obtain a list of candidate text character shapes by using AdaBoost to detect text regions followed by binarization to detect candidate text character boundaries within text regions. This list is represented by a set of particles which are weighted by the similarity to the deformable templates for text characters (see below):

$$S_{1r}(W) = \{\ (z_{1r}^{(\mu)}, \omega_{1r}^{(\mu)}) :\ \mu = 1, 2, ..., N_{1r}\}.$$

Similarly, we specify another set of weighted particles for removing text characters:

$$S_{1l}(W') = \{\ (z_{1l}^{(\nu)}, \omega_{1l}^{(\nu)}) :\ \nu = 1, 2, ..., N_{1l}\}.$$

$\{z_{1r}^{(\mu)}\}$ and $\{z_{1l}^{(\nu)}\}$ represent the possible (discretized) shape positions and text character deformable templates for creating or removing text, and $\{\omega_{1r}^{(\mu)}\}$ and $\{\omega_{1l}^{(\nu)}\}$ are their corresponding weights. The particles are then used to compute proposal probabilities

$$\mathbf{Q}_{1r}(W'|W:\mathbf{I}) = \frac{\omega_{1r}(W')}{\sum_{\mu=1}^{N_{1r}} \omega_{1r}^{(\mu)}}, \quad \mathbf{Q}_{1l}(W|W',\mathbf{I}) = \frac{\omega_{1l}(W)}{\sum_{\nu=1}^{N_{1l}} \omega_{1l}^{(\nu)}}.$$

The weights $\omega_{1r}^{(\mu)}$ and $\omega_{1l}^{(\nu)}$ for creating new text characters are specified by shape affinity measures, such as shape contexts. For deleting text characters we calculate $\omega_{1l}^{(\nu)}$ directly from the likelihood and prior on the text character. Ideally these weights will approximate the ratios $\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})}$ and $\frac{p(W|\mathbf{I})}{p(W'|\mathbf{I})}$.

5.2.2 Sub-kernel II: birth and death of face

The sub-kernel for the birth and death of faces is very similar to the sub-kernel of birth and death of text. We use AdaBoost to detect candidate faces. Face boundaries are obtained directly from using edge detection to give candidate face boundaries. The proposal probabilities are computed similarly to those for sub-kernel I.

5.2.3 Sub-kernel III: splitting and merging regions

This pair of jumps is used to create or delete nodes by splitting and merging regions (nodes). We start with a parse graph W and transition into parse graph W' by splitting node i into nodes j and k. Conversely, we transition back to W by merging nodes j and k into node i. The selection of which region i to split is based on a robust function on $p(\mathbf{I}_{R_i}|\zeta_i, L_i, \Theta_i)$ (i.e. the worse the model for region R_i fits the data, the more likely we are to split it). For merging, we use a region affinity measure and propose merges between regions which have high affinity.

Formally, we define W, W':

$$W = (K, (\zeta_k, L_k, \Theta_k), W_-) \rightleftharpoons W' = (K+1, (\zeta_i, L_i, \Theta_i), (\zeta_j, L_j, \Theta_j), W_-)$$

where W_- denotes the attributes of the remaining $K-1$ nodes in the graph.

We obtain proposals by seeking approximations to equation (**??**) as follows.

We first obtain three edge maps. These are given by Canny edge detectors at different scales. We use these edge maps to create a list of particles for splitting $S_{3r}(W)$. A list of particles for merging is denoted by $S_{3l}(W')$.

$$S_{3r}(W) = \{\ (z_{3r}^{(\mu)}, \omega_{3r}^{(\mu)}) : \ \mu = 1, 2, ..., N_{3r}.\}, \quad S_{3l}(W') = \{\ (z_{3l}^{(\nu)}, \omega_{3l}^{(\nu)}) : \ \nu = 1, 2, ..., N_{3l}.\}$$

where $\{z_{3r}^{(\mu)}\}$ and $\{z_{3l}^{(\nu)}\}$ represent the possible (discretized) positions for splitting and merging, and their weights $\{\omega_{3r}\}, \{\omega_{3l}\}$ will be defined shortly. In other words, we can *only* split a region i into regions j and k along a contour z_{3r}^{μ} (i.e. z_{3r}^{μ} forms the new boundary). Similarly we can only merge regions j and k into region i by deleting a boundary contour z_{3l}^{μ}.

We now define the weights $\{\omega_{3r}\}, \{\omega_{3l}\}$. These weights will be used to determine probabilities for the splits and merges by:

$$\mathbf{Q}_{3r}(W'|W : \mathbf{I}) = \frac{\omega_{3r}(W')}{\sum_{\mu=1}^{N_{3r}} \omega_{3r}^{(\mu)}}, \quad \mathbf{Q}_{3l}(W|W' : \mathbf{I}) = \frac{\omega_{3l}(W)}{\sum_{\nu=1}^{N_{3l}} \omega_{3l}^{(\nu)}}.$$

Again, we would like ω_{3r}^{μ} and ω_{3l}^{ν} to approximate the ratios $\frac{p(W|\mathbf{I})}{p(W'|\mathbf{I})}$ and $\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})}$ respectively. $\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})}$ is given by:

$$\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})} = \frac{p(\mathbf{I}_{R_i}|\zeta_i, L_i, \Theta_i) p(\mathbf{I}_{R_j}|\zeta_j, L_j, \Theta_j)}{p(\mathbf{I}_{R_k}|\zeta_k, L_k, \Theta_k)} \cdot \frac{p(\zeta_i, L_i, \Theta_i) p(\zeta_j, L_j, \Theta_j)}{p(\zeta_k, L_k, \Theta_k)} \cdot \frac{p(K+1)}{p(K)}$$

This is expensive to compute, so we approximate $\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})}$ and $\frac{p(W|\mathbf{I})}{p(W'|\mathbf{I})}$ by:

$$\omega_{3r}^{(\mu)} = \frac{q(R_i, R_j)}{p(\mathbf{I}_{R_k}|\zeta_k, L_k, \Theta_k)} \cdot \frac{[q(L_i) q(\zeta_i, \Theta_i)][q(L_j) q(\zeta_j, \Theta_j)]}{p(\zeta_k, L_k, \Theta_k)}. \tag{17}$$

$$\omega_{3l}^{(\nu)} = \frac{q(R_i, R_j)}{p(\mathbf{I}_{R_i}|\zeta_i, L_i, \Theta_i) p(\mathbf{I}_{R_j}|\zeta_j, L_j, \Theta_j)} \cdot \frac{q(L_k) q(\zeta_k, \Theta_k)}{p(\zeta_i, L_i, \Theta_i) p(\zeta_j, L_j, \Theta_j)}, \tag{18}$$

Where $q(R_i, R_j)$ is an *affinity measure* of the similarity of the two regions R_i and R_j (it is a weighted sum of the intensity difference $|\bar{I}_i - \bar{I}_j|$ and the chi-squared difference between the intensity histograms), $q(L_i)$ is given by the priors on the shape descriptors, and $q(\zeta_i, \Theta_i)$ is obtained by clustering in parameter space.

5.2.4 Jump II: Switching Node Attributes

These moves switch the attributes of a node i. This involves changing the region type ζ_i and the model parameters Θ_i.

The move transitions between two states:

$$W = ((\zeta_i, L_i, \Theta_i), W_-) \rightleftharpoons W' = ((\zeta_i', L_i', \Theta_i'), W_-)$$

The proposal should approximate:

$$\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})} = \frac{p(\mathbf{I}_{R_i}|\zeta_i', L_i', \Theta_i')p(\zeta_i', L_i', \Theta_i')}{p(\mathbf{I}_{R_i}|\zeta_i, L_i, \Theta_i)p(\zeta_i, L_i, \Theta_i)}.$$

We approximate this by a weight $\omega_4^{(\mu)}$ given by

$$\omega_4^{(\mu)} = \frac{q(L_i')q(\zeta_i', \Theta_i')}{p(\mathbf{I}_{R_i}|\zeta_i, L_i, \Theta_i)p(\zeta_i, L_i, \Theta_i)},$$

where $q(L_i')q(\zeta_i', \Theta_i')$ are the same functions used in the split and merge moves. The proposal probability is the weight normalized in the candidate set, $\mathbf{Q}_4(W'|W:\mathbf{I}) = \frac{\omega_4(W')}{\sum_{\mu=1}^{N_4}\omega_4^{(\mu)}}$.

Appendix: Markov Chain kernels and sub-kernels

For each move we define a Markov Chain sub-kernel by a transition matrix $\mathcal{K}_a(W'|W:\mathbf{I})$ with $a \in \mathcal{A}$ being an index. This represents the probability that the system makes a transition from state W to state W' when sub-kernel a is applied (i.e. $\sum_{W'} \mathcal{K}_a(W'|W:\mathbf{I}) = 1,\ \forall\ W$). Kernels which alter the graph structure are grouped into reversible pairs. For example, the sub-kernel for node creation $\mathcal{K}_{a,r}(W'|W:\mathbf{I})$ is paired with the sub-kernel for node deletion $\mathcal{K}_{a,l}(W'|W:\mathbf{I})$. This can be combined into a paired sub-kernel $\mathcal{K}_a = \rho_{ar}\mathcal{K}_{a,r}(W'|W:\mathbf{I}) + \rho_{al}\mathcal{K}_{a,l}(W'|W:\mathbf{I})$ $(\rho_{ar} + \rho_{al} = 1)$. This pairing ensures that $\mathcal{K}_a(W'|W:\mathbf{I}) = 0$ if, and only if, $\mathcal{K}_a(W|W':\mathbf{I}) = 0$ for all states $W, W' \in \Omega$. The sub-kernels (after pairing) are constructed to obey the detailed balance condition:

$$p(W|\mathbf{I})\mathcal{K}_a(W'|W:\mathbf{I}) = p(W'|\mathbf{I})\mathcal{K}_a(W|W':\mathbf{I}). \tag{19}$$

The full transition kernel is expressed as:

$$\mathcal{K}(W'|W:\mathbf{I}) = \sum_a \rho(a:\mathbf{I})\mathcal{K}_a(W'|W:\mathbf{I}), \quad \sum_a \rho(a:\mathbf{I}) = 1, \quad \rho(a:\mathbf{I}) > 0. \tag{20}$$

To implement this kernel, at each time step the algorithm selects the choice of move with probability $\rho(a:\mathbf{I})$ for move a, and then uses kernel $\mathcal{K}_a(W'|W;\mathbf{I})$ to select the transition from state W to state W'. Note that both probabilities $\rho(a:\mathbf{I})$ and $\mathcal{K}_a(W'|W;\mathbf{I})$ depend on the input image $\mathbf{I}$.

The full kernel obeys detailed balance, equation (19), because all the sub-kernels do. It will also be ergodic, provided the set of moves is sufficient (i.e. so that we can transition between any two states $W, W' \in \Omega$ using these moves). These two conditions ensure that $p(W|\mathbf{I})$ is the invariant (target) probability of the Markov Chain.

Applying the kernel $\mathcal{K}_{a(t)}$ updates the Markov chain state probability $\mu_t(W)$ at step t to $\mu_{t+1}(W')$ at $t+1$:

$$\mu_{t+1}(W') = \sum_W \mathcal{K}_{a(t)}(W'|W:\mathbf{I})\mu_t(W). \tag{21}$$

In summary, the DDMCMC image parser simulates a Markov chain $\mathcal{MC}$ with a unique invariant probability $p(W|\mathbf{I})$. At time t, the Markov chain state (i.e. the parse graph) W follows a probability μ_t which is the product of the sub-kernels selected up to time t,

$$W \sim \mu_t(W) = \nu(W_o) \cdot [\mathcal{K}_{a(1)} \circ \mathcal{K}_{a(2)} \circ \cdots \circ \mathcal{K}_{a(t)}](W_o, W) \ \longrightarrow\ p(W|\mathbf{I}). \tag{22}$$

where $a(t)$ indexes the sub-kernel selected at time t. As the time t increases, $\mu_t(W)$ approaches the posterior $p(W|\mathbf{I})$ monotonically at a geometric rate independent of the starting configuration. The following convergence theorem is useful for image parsing because it helps quantify the effectiveness of the different sub-kernels.

Theorem 1 *The Kullback-Leibler divergence between the posterior $p(W|\mathbf{I})$ and the Markov chain state probability decreases monotonically when a sub-kernel $\mathcal{K}_{a(t)}, \forall\ a(t) \in \mathcal{A}$ is applied,*

$$KL(p(W|\mathbf{I})\,||\,\mu_t(W)) - KL(p(W|\mathbf{I})\,||\,\mu_{t+1}(W)) \geq 0 \tag{23}$$

The decrease of KL-divergence is strictly positive and is equal to zero only after the Markov chain becomes stationary, i.e. $\mu = p$.

The theorem is related to the second law of thermodynamics, and its proof makes use of the detailed balance equation (19). This KL divergence gives a measure of the "power" of each sub-kernel $\mathcal{K}_{a(t)}$ and so it suggests an efficient mechanism for selecting the sub-kernels at each time step, see Section (**??**). By contrast, classic convergence analysis shows that the convergence of the Markov Chain is exponentially fast, but does not give measures of power of sub-kernels.

Proof. For notational simplicity, we ignore the dependencies on the kernels and probabilities on the input image **I**.

Let $\mu_t(W_t)$ be the state probability at time step t. After applying a sub-kernel $\mathcal{K}_a(W_{t+1}|W_t)$, its state becomes W_{t+1} with probability:

$$\mu_{t+1}(W_{t+1}) \ = \sum_{W_t} \mu_t(W_t)\mathcal{K}_a(W_{t+1}|W_t). \tag{24}$$

The joint probability $\mu(W_t, W_{t+1})$ for W_t and W_{t+1} can be expressed in two ways as:

$$\mu(W_t, W_{t+1}) = \mu_t(W_t)\mathcal{K}_a(W_{t+1}|W_t) = \mu_{t+1}(W_{t+1})p_{\mathcal{MC}}(W_t|W_{t+1}), \tag{25}$$

where $p_{\mathcal{MC}}(W_t|W_{t+1})$ is the "posterior" probability for state W_t at time step t conditioned on state W_{t+1} at time step $t+1$.

This joint probability $\mu(W_t, W_{t+1})$ can be compared to the joint probability $p(W_t, W_{t+1})$ at equilibrium (i.e. when $\mu_t(W_t) = P(W_t)$). We can also express $p(W_t, W_{t+1})$ in two ways:

$$p(W_t, W_{t+1}) = p(W_t)\mathcal{K}_a(W_{t+1}|W_t) = p(W_{t+1})\mathcal{K}_a(W_{t+1}, W_t), \tag{26}$$

where the second equality is obtained using the detailed balance condition on $\mathcal{K}_a$.

We calculate the Kullback-Leibler (K-L) divergence between the joint probabilities $P(W_t, W_{t+1})$ and $\mu(W_t, W_{t+1})$ in two ways using the first and second equalities in equations (25,26). We obtain two expressions (1 & 2) for the K-L divergence:

$$\begin{aligned}
& KL(p(W_t, W_{t+1})\,||\,\mu(W_t, W_{t+1})) = \sum_{W_{t+1}}\sum_{W_t} p(W_t, W_{t+1}) \log \frac{p(W_t, W_{t+1})}{\mu(W_t, W_{t+1})} \\
\overset{1}{=} \quad & \sum_{W_t} p(W_t) \sum_{W_{t+1}} \mathcal{K}_a(W_{t+1}|W_t)) \log \frac{p(W_t)\cdot\mathcal{K}_a(W_{t+1}|W_t)}{\mu_t(W_t)\mathcal{K}_a(W_{t+1}|W_t)} \\
= \quad & KL(p(W)\,||\,\mu_t(W)) && (27) \\
\overset{2}{=} \quad & \sum_{W_{t+1}}\sum_{W_t} \mathcal{K}_a(W_t|W_{t+1}))p(W_{t+1}) \log \frac{p(W_{t+1})\mathcal{K}_a(W_t|W_{t+1})}{\mu_{t+1}(W_{t+1})p_{\mathcal{MC}}(W_t|W_{t+1})} \\
= \quad & KL(p(W_{t+1})\,||\,\mu_{t+1}(W)) + E_{p(W_{t+1})}[KL(\mathcal{K}_a(W_t|W_{t+1})||p_{\mathcal{MC}}(W_t|W_{t+1}))] && (28)
\end{aligned}$$

We equate the two alternatives expressions for the KL divergence, using equations (27) and (28), and obtain

$$KL(p(W)\,||\,\mu_t(W)) - KL(p(W)\,||\,\mu_{t+1}(W)) = E_{p(W_{t+1})}[KL(\mathcal{K}_{a(t)}(W_t|W_{t+1})||p_{\mathcal{MC}}(W_t|W_{t+1}))]$$

This proves that the K-L divergence decreases monotonically. The decrease is zero only when $\mathcal{K}_{a(t)}(W_t|W_{t+1}) = p_{\mathcal{MC}}(W_t|W_{t+1})$ (because $KL(p||\mu) \geq 0$ with equality only when $p = \mu$). This occurs if and only if $\mu_t(W) = p(W)$ (using the definition of $p_{\mathcal{MC}}(W_t|W_{t+1})$, given in equation (25), and the detailed balance conditions for $\mathcal{K}_{a(t)}(W_t|W_{t+1})$). [End of Proof].