Graphical Epitome Processing

by

Vincent Cheung

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

Graphical Epitome Processing

Vincent Cheung

Doctor of Philosophy

Graduate Department of Electrical and Computer Engineering

University of Toronto

2013

This thesis introduces principled, broadly applicable, and efficient patch-based models for data processing applications. Recently, "epitomes" were introduced as patch-based probability models that are learned by compiling together a large number of examples of patches from input images. This thesis describes how epitomes can be used to model video data and a significant computational speedup is introduced that can be incorporated into the epitome inference and learning algorithm. In the case of videos, epitomes are estimated so as to model most of the small space-time cubes from the input data. Then, the epitome can be used for various modelling and reconstruction tasks, of which we show results for video super-resolution, video interpolation, and object removal. Besides computational efficiency, an interesting advantage of the epitome as a representation is that it can be reliably estimated even from videos with large amounts of missing data. This ability is illustrated on the task of reconstructing the dropped frames in a video broadcast using only the degraded video. Further, a new patch-based model is introduced, that when applied to epitomes, accounts for the varying geometric configurations of object features. The power of this model is illustrated on tasks such as multiple object registration and detection and missing data interpolation, including a difficult task of photograph relighting.

Acknowledgements

I would like to thank my thesis advisor, Brendan Frey, for all the opportunities that he has provided me while working in his group and his continued support in all my endeavours. I would like to thank Nebojsa Jojic for believing in me and pushing me to excel. Anitha Kannan taught me the ropes, to which I am grateful to her for. Thanks also goes out to the rest of the Probabilistic and Statistical Inference Group for their support and feedback. Finally, my gratitude goes out to my friends and family for their support throughout my graduate career.

Contents

Chapter 1

Introduction

Patch models create a representation of a category of input data by creating a set of patches that represent all of the constituent parts of the given category of input data. For example, a category of input data can be a particular kind of images, such as images of human faces. A significant advantage of patch models is that each constituent part of a category of data needs only one patch. Thus, because there will often be a large amount of repetition within the data in a category, a patch model can be much smaller than the data it is representing and still capture every aspect of the represented category of data. Because of the benefits provided by patch models, the use of such models is increasing in popularity. Specifically, patch models for images have recently seen increased use for various low level image processing tasks [8, 11, 13, 14, 15, 19, 21, 23, 24, 30, 34, 38, 40, 42].

This thesis introduces principled, broadly applicable, and efficient patch-based models for data processing applications. These models work across many different applications, including novel ones, even without any domain knowledge because of their principled probability models that require minimal parameter tuning. While this thesis shows results for a number of applications, the results are primarily to demonstrate the broad applicability of the models, which are able to be applied to these varied tasks by optimizing under their probability models. This thesis is not focused on any one particular

application, but rather, the primary contribution is the models themselves.

Recently, "epitomes" were introduced as patch-based probability models that are learned by compiling together a large number of examples of patches from input images [25]. The image epitome model is appealing in that its principled generative model allows for various modelling and reconstruction tasks. While powerful, the model is lacking in some aspects that restrict its practical usage. This thesis extends the epitome model in several ways and proposes it as a novel patch model platform for analyzing visual data and performing various tasks with the data.

A background on patch models and the image epitome model is given in Chapter 2. Two of the key parameters of the epitome model are the epitome size and patch size, which have dramatic effects on the ability of the epitome to model the data, yet prior work has not provided any guidance in terms of selecting these parameters. In Chapter 3, these parameters are studied using natural image statistics to gain a better understanding of them. Learning and inference with epitome models is a computationally intensive procedure, so in Chapter 4, a novel, efficient algorithm is introduced that dramatically reduces the computational complexity of the patch comparisons necessary in the epitome model. The epitome model was originally presented as a model for image data. In Chapter 5, the model is extended to videos by using 3D patches from the video, while also presenting a novel way to model missing data. The use of the extended model is then applied to applications of video super-resolution, video interpolation, object removal, and denoising.

The epitome model, like other patch models, capture local correlations between pixels in a patch. Thus, if a patch in a first image matches well with a patch in a second image, then a second patch in the first image that shares pixels with the first patch should also match well to a similarly displaced second patch in the second image. Conventional image processing applications can use these local correlations to piece or cluster together groups of patches to form textures that can then be used to process portions of an image.

However, conventional patch models have difficulty in capturing longer range correlations between more distant data elements in a set of data. Because of this, conventional patch models are not able to determine the overall context a category of data on their own. For example, while a conventional patch model can represent small portions of an image based on local correlations between pixels within the image that are close together, the patch model cannot use these portions to create an entire image because it lacks the overall context of how the image is constructed. Thus, in applications that utilize these patch models, this context must be provided manually. Some conventional patch models have attempted to add some degree of context information by utilizing larger patches but this approach is not fully effective. For example, larger patches are more difficult to match to data than smaller patches, which means that more patches are required to represent a category of data. Because using larger patches also requires the use of more patches, using larger patches can considerably increase the required size of a patch model. Additionally, patch models that utilize larger patches are still not able to capture the entire overall context of an image because, like patch models that utilize smaller patches, they can capture only local correlations between pixels within a patch.

Thus, in view of at least the above, there exists a need in the art for a patch model that is able to capture the entire overall context of a category of data without an undue sacrifice in size. Such a model is introduced in Chapter 6, which is applied to both epitomes and an image illumination manipulation application.

Chapter 2

Background

2.1 Patch Based Modeling

The technique of using small image patches to account for high-order statistics in image and video data continues to grow in popularity in the vision community. A patch in an image is defined as a set of neighbouring values that are presumed to be related because of their close proximity. In video, the concept is the same except that patches take on a 3D shape, where two of the dimensions are spatial and the third is time. The reason for the prevalent use of patches is their ability to capture high-order statistics and model short range dependencies in a computationally efficient manner. One of the first uses of patches was to find corresponding points in neighboring video frames for computing optical flow. Instead of simply matching patches, Jepson and Black introduced probabilistic patch models that account for outliers [24]. Soon after that, Wang and Adelson showed that patches could be used for efficient video compression [38]. In related work, "textons" use image patches within a structured mathematical framework, to account for texture using a patch-based representation [42]. In these cases, patches were used primarily for analyzing image data.

More recently, patches have been used successfully for analyzing and synthesizing

images and videos. Patches from one part of an image have been stitched together to synthesize new images with similar texture, or to in-paint texture into an interior region [11, 13, 14]. This approach has also been used to fill in missing or occluded regions of video data [40]. Image denoising has been performed by averaging patches of similar intensity [8]. Libraries of patches derived from high-resolution images have been used for super-resolution, both in static images and video sequences [6, 16]. Patch-based image models have been used for the purpose of "texture transfer", also known as "unsupervised image analogies" [21, 34]. Patches have also been used for object and class recognition [15, 30], tracking [23], and stereo [19].

To jointly analyze and synthesize data, patch-based probability models were introduced in [25]. These models, called "epitomes", compile patches drawn from input images into a condensed image model that represents many of the high-order statistics in the input image. An advantage of learning a patch-based model from the input data is that the model is broadly applicable and can be used for a variety of inference tasks, such as image/texture classification, in-painting, super-resolution, texture transfer, and image segmentation [25]. Epitomes have found uses in other application areas, including speech [28], motion capture data [29], and molecular biology [26].

2.2 Image Epitome Model

The epitome of an image is a miniature, condensed version that accurately accounts for the interesting properties of the image. Image epitomes have a variety of data processing applications, such as object detection, texture segmentation, and image retrieval. An example of an epitome is shown in Fig. 2.1. The epitome is 16 times smaller than the image, yet still contains the necessary elements to reconstruct the image.

The epitome of an image can be considered to be a generative model of image patches taken from the image. Let there be a set of M patches taken from the original image

(a) (b)

Figure 2.1: A 128x128 image (a) and its 32x32 epitome (b).

giving patches $\{\boldsymbol{Z}_k\}_{k=1}^M$, each containing pixels from a subset of image coordinates S_k. The patch size need not be fixed and the image patches can overlap. Each patch $\boldsymbol{Z}_k$, is generated by using a hidden mapping $\mathcal{T}_k$ that map co-ordinates from the epitome $\boldsymbol{e}$ to the patch, as described by the Bayesian network [17] shown in Fig. 2.2. The joint

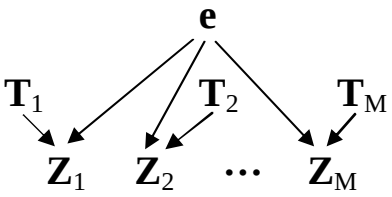

Figure 2.2: The generative patch model epitome Bayesian network.

probability distribution of this network can be written as:

$$p(\{\boldsymbol{Z}_k, \mathcal{T}_k\}_{k=1}^M, \boldsymbol{e}) = p(\boldsymbol{e}) \prod_{k=1}^{M} p(\mathcal{T}_k) p(\boldsymbol{Z}_k | \mathcal{T}_k, \boldsymbol{e}) \tag{2.1}$$

The epitome is represented by a mean and variance, $\boldsymbol{e} = (\mu, \phi)$ and the likelihood is modelled as being normally distributed over the pixels, $z_{i,k}$, with mean $\mu_{\mathcal{T}_k(i)}$ and variance $\phi_{\mathcal{T}_k(i)}$:

$$p(\boldsymbol{Z}_k | \mathcal{T}_k, \boldsymbol{e}) = \prod_{i \in S_k} \mathcal{N}(z_{i,k}; \mu_{\mathcal{T}_k(i)}, \phi_{\mathcal{T}_k(i)}) \tag{2.2}$$

Learning involves the estimation of the posterior distribution, $p(\mathcal{T}_k, \boldsymbol{e} | \boldsymbol{Z}_k)$. To simplify learning, an independence of the epitome and hidden mappings in the posterior can be

assumed:

$$\begin{aligned} p(\{\mathcal{T}_k\}_{k=1}^M, \boldsymbol{e}|\{\boldsymbol{Z}_k\}_{k=1}^M) &\approx q(\boldsymbol{e})q(\{\mathcal{T}_k\}_{k=1}^M) \\ &\approx q(\boldsymbol{e})\prod_{k=1}^M q(\mathcal{T}_k) \end{aligned} \tag{2.3}$$

Learning is done in this model using a variational expectation maximum (EM) algorithm [17] and the posterior is updated iteratively by:

$$q(\mathcal{T}_k) \propto \prod_{i \in S_k} \mathcal{N}(z_{i,k}; \mu_{\mathcal{T}_k(i)}, \phi_{\mathcal{T}_k(i)}) \tag{2.4}$$

The update equation for the epitome parameters are given by:

$$\mu_j \leftarrow \frac{\sum_k \sum_{i \in S_k} \sum_{\mathcal{T}_k, \mathcal{T}_k(i)} q(\mathcal{T}_k) z_{i,k}}{\sum_k \sum_{i \in S_k} \sum_{\mathcal{T}_k, \mathcal{T}_k(i)} q(\mathcal{T}_k)} \tag{2.5}$$

$$\phi_j \leftarrow \frac{\sum_k \sum_{i \in S_k} \sum_{\mathcal{T}_k, \mathcal{T}_k(i)} q(\mathcal{T}_k)(z_{i,k} - \mu_j)^2}{\sum_k \sum_{i \in S_k} \sum_{\mathcal{T}_k, \mathcal{T}_k(i)} q(\mathcal{T}_k)} \tag{2.6}$$

Learning under this generative patch model of the epitome is computationally dominated by this approximation of the posterior distribution in the expectation, or E, step. It can be shown that up to an additive constant, the log-posterior over the mappings can be written as

$$\log q(\mathcal{T}_k) = \frac{1}{2}\sum_{i \in S_k} \log \phi_{\mathcal{T}_k(i)} - \frac{1}{2}\sum_{i \in S_k} (z_{i,k} - \mu_{\mathcal{T}_k(i)})^2 / \phi_{\mathcal{T}_k(i)} \tag{2.7}$$

The first summation in this equation does not depend on the given patch, $\boldsymbol{Z}_k$ and can be computed with a low order of complexity. However, the second term is dependent on both the image patch and the epitome. This summation is essentially the Euclidean distance between the patch $\boldsymbol{Z}_k$ and the patch in the epitome given by the mapping $\mathcal{T}_k$, where each squared difference is weighted by $1/\phi_{\mathcal{T}_k(i)}$. In learning, it is required to compute this distance between all $\boldsymbol{Z}_k$ patches from the original image and all patches in the epitome, a computationally intensive procedure. Inference under this model involves maximizing (2.7) over all possible mappings, $\mathcal{T}_k$.

Chapter 3

Epitome Parameters

3.1 Introduction

The epitome model requires that the epitome size and patch size be specified; however, it provides no guidance as to the selection of these parameters. Both parameters have a dramatic effect on the features that the epitome is able to model and hence, affect the success of using epitome models for applications. We examine both using natural image statistics to see how changing these parameters affect the modelling capabilities of epitomes.

3.1.1 Epitome Size

The image epitome [25] is a compact representation of an image that has many of the same features as the original image. One of the most significant open questions related to epitomes is determining the size of the epitome. The epitome's size determines the amount of compression of the epitome and affects the results of the use of the epitome. Fig. 3.1 shows an image and epitomes of three different sizes. Each epitome captures as many features as it can within the available resources. The epitome is taken over a torus, that is, the epitome is circularly wrapped along the edges in order to best make

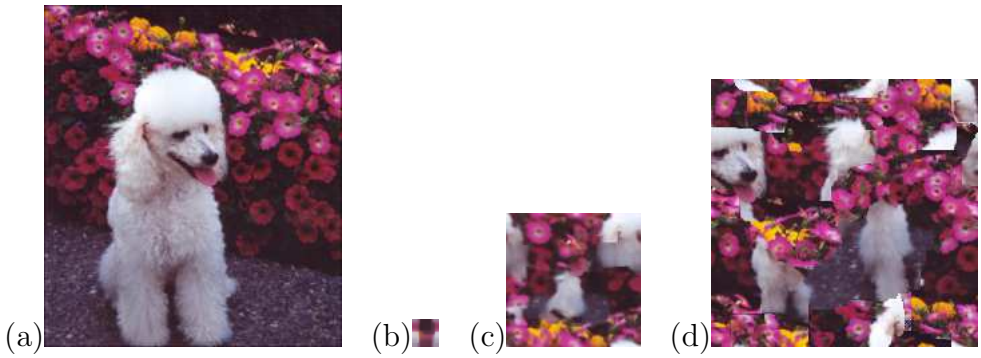

Figure 3.1: (a) A 223x256 image and its (b) 20x20, (c) 100x100, and (d) 200x200 epitome.

use of its resources. While the 20x20 epitome has very little real estate, it still manages to model the basic colours in the image. Conversely, the 200x200 epitome is overly large and is able to model nearly all the features. The interesting cases are in between. The 100x100 epitome is able to capture many of the image's features while being nearly six times smaller than the original image. Note how the epitome has a few flowers that are representative of the many flowers in the original image. The epitome captures many of the high-order statistics, but in a local manner. Entire objects and textures are intact in the epitome and generally, interactions between features in the epitome are reflective of those in the original image, eg. dog fur next to concrete, but global trends are not generally maintained.

The epitome provides a unique ability to compare statistics since it itself is an image, which is not the case with such approaches as vector quantization. Since the epitome itself models the statistics of the image, the idea of this chapter is to study the statistics of the image epitome and compare them to those of the original image. Natural images have statistics that differ significantly from random images; natural image statistics generally have heavy-tailed distributions, as opposed to Gaussian. The image epitome is far from random, but it purposely changes the statistics as it compresses the image's features. This analysis of the image epitome can give us insight into the ability of the epitome model to maintain the original image statistics and can aid in determining the size of the

epitome.

3.1.2 Patch size

The choice of the patch size used within the epitome model has a significant affect on the resulting epitome. The size of the patch determines what dependencies between pixels are modeled. If the patch size is reduced to a single pixel, then the "patch" comparisons will completely lack context and the resulting epitome would be a set of independent values. Increasing the patch size helps to capture some of the short-range dependencies by tying some of the epitome values together, but it only reduces, not eliminates, the problems encountered by just using single pixels. The easiest way to incorporate longer correlations is to simply use larger patches. The use of larger patches may however, lead to undesired results because the patches would capture too much information. In the extreme example of where the patch size is the same size as the entire image, the "patch" comparison is unable to accommodate invariances such as rearranged or repeated textures.

The size of the patch is thus a balance to capture as many dependencies as possible without capturing irrelevant ones. However, the assumption that there is one single patch size that is appropriate over an entire image or video may not be true because of the large variation in visual data. In textureless areas, it would be best to have a large patch size to capture many of the short range dependencies of the data; however, in other areas where the features are small or there are object boundaries, the patch size must be correspondingly small.

The typical solution to the patch size problem at least with iterative algorithms is to use multiple patch sizes, starting with large patches and decreasing the patch size over the course of the iterations. Large patches capture larger features and longer range correlations, while smaller patch sizes better match local areas. While it is often not explicitly mentioned that multiple patch sizes are used, it is often implicitly the case as

many algorithms make use of Gaussian pyramids (see Sec. 4.4.2) for computational reasons. In this framework, when executing the algorithm on blurred, sub-sampled versions of the data with the same patch size used throughout all of the pyramid levels, a larger patch size is essentially being used initially.

Other algorithms have made limited attempts to dynamically adapt the patch size by subdividing the patch dyadically [25, 33]. There are some limitations to how much large patches can accomplish, in that the correlations must be enclosed in some reasonable rectangular region and cannot be separated by large areas. Chapter 6 describes a principled approach that captures the desired dependencies by using relatively small patches and explicitly modeling longer range correlations between patches.

3.2 Evaluation Criteria

Epitome parameters are dependent on the application. In general, the epitome is used to model features in the input data for use in applications, be it compression or as a regularizer.

The epitome of an image is interesting in that it is itself an image and thus, the ideas of statistics of natural images can be applied. It is well established that natural images have statistics that are significantly different than random images. Images generally have heavy-tailed distributions and these can aid in determining the quality of epitomes. Several statistics are explored in this chapter. There are however, a few requirements on the statistics in order to be able to successfully compare the original image and the epitome. The epitome aims to remove redundancy and purposely changes the overall statistics of the image in order to reduce its size. Further, the global image statistics are not maintained. ICA and PCA approaches were not found to be particularly suitable here since the most significant components in the image are affected by the compression of the epitome model and it was difficult to compare the epitome to the input image since

the components found were not identical. The epitome does not maintain the spatial location or geometry of features. Further, the epitome is not of the same size as the original image. The measured statistics need to be invariant to these transformations in order to be effective. The rest of this section describes the single pixel, derivative, bandpass filter, and wavelet statistics used in this chapter.

3.2.1 Single pixel statistics

The easiest, yet often effective, statistic that can be computed on an image is the distribution of the pixel values in the image. Since the epitome of an image is supposed to contain the same features as the original image, it should also have a similar distribution of pixel values. It is not however, expected that the distributions would be identical since the epitome compresses the image in a highly non-linear fashion by exploiting the redundancy in the image. For example, in Fig. 3.1c, the original image contains numerous flowers, but the epitome contains only a representative few. Thus, while the epitome's pixel distribution contains these values for the flowers, they are not in the same proportions as in the original image.

Since the images used in this chapter have colour, it is desired to have a representation of the pixel distribution that reflects this fact. In an analogous fashion to one-dimensional histograms and similar to [37], colour histograms are formed by binning the red, green, and blue components and tracking counts of pixels in each bin. Fig. 3.2 shows a schematic representation of this colour histogram. This 3D histogram is applied to all of the statistics.

3.2.2 Derivative statistics

While it is important for the epitome to capture a similar distribution of pixel values as the original image, another important characteristic is to model the horizontal and vertical derivatives. With too small of an epitome, too much averaging occurs and the

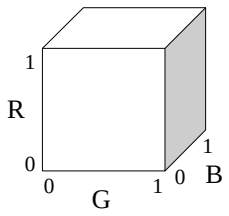

Figure 3.2: Axes of a colour histogram with the red, green, and blue components scaled to between 0 and 1.

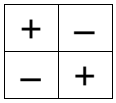

Figure 3.3: A local bandpass filter.

features are blended together leaving blurred edges and thus, the image derivatives would not have the same distribution as the original image. An epitome should be large enough so that it has similar feature edges as the input image and consequently, a similar distribution of derivatives. 3D histograms of the image derivatives allow more of a structural comparison than the pixel statistics while still maintaining the ability to compare the epitome with the original image.

3.2.3 Filter statistics

A simple linear filtering is considered here of the same form as from [35]. Fig. 3.3 shows the 2x2 bandpass filter used. This particular filter passes no signal at zero spatial frequency. Histograms of this filter's output has been shown to have the characteristic heavy-tail distribution.

3.2.4 Wavelet statistics

Finally, joint statistics of the pixels are examined in the wavelet domain. The 2D Haar wavelet is shown in [22] to give interesting distributions over natural images. Fig. 3.4 shows the four Haar filters. Each 2x2 filter is applied in an non-overlapping manner. A

+1/2	+1/2
-1/2	-1/2

Horizontal

+1/2	-1/2
+1/2	-1/2

Vertical

+1/2	-1/2
-1/2	+1/2

Diagonal

+1/2	+1/2
+1/2	+1/2

Low Pass

Figure 3.4: Haar wavelet filters.

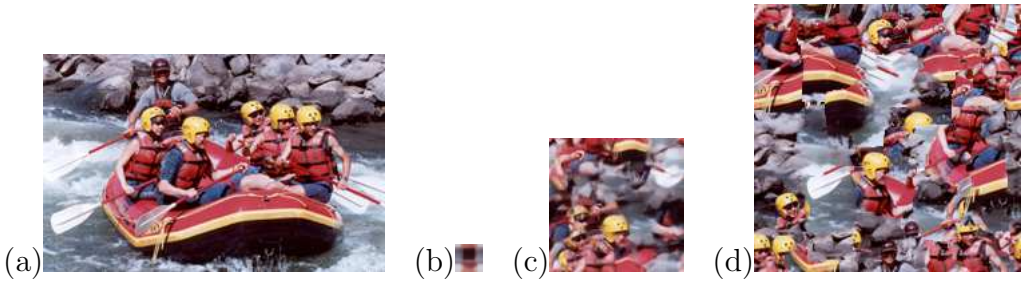

Figure 3.5: (a) A 256x163 image and its (b) 20x20, (c) 100x100, and (d) 200x200 epitome.

multi-resolution analysis is achieved by recursively applying this procedure upon the low pass filter response until a 1x1 image is obtained. Histograms of each of the horizontal, vertical, and diagonal responses is used as a basis to evaluate the epitome.

3.3 Experiments

The experiments performed aimed to gauge the ability of the image epitome to model the statistics of the original image and use this to aid in determining an appropriate size of the epitome. For both the images in Fig. 3.1 and Fig. 3.5, epitomes of size 20x20 - 200x200, in increments of 10 pixels along each dimension, were computed. In all cases patches of size 32x32, 16x16, and 8x8 were used. The larger epitomes would have had greater spatial regularity had larger patch sizes been used, but the patches remained fixed for all sizes for comparison purposes.

The analysis proceeds by computing the statistics from Sec. 3.2 upon the original

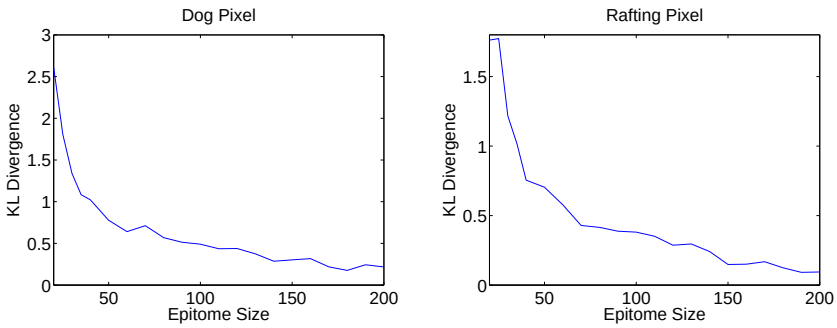

Figure 3.6: Pixel value distribution comparison.

image and comparing them to those of the epitomes. The empirical colour histograms are compared using the Kullback-Leibler divergence, which evaluates a distance measure between two distributions, p and q,

$$\mathrm{d}(p \,\|\, q) = \sum_{(r,g,b)} p(r,g,b) \log \frac{p(r,g,b)}{q(r,g,b)}. \tag{3.1}$$

For the 3D colour histograms, the sums are taken over the bins (r, g, b). The KL-divergence is 0 when $p = q$. The epitome distribution is taken to be p and the original distribution is q so that when $q(r, g, b) = 0$, $p(r, g, b)$ must also be 0 for finite distance. Since the comparison measure is based on empirical histograms, there is some sensitivity to the resolution of bins. It was found that the results are fairly resilient to the number of bins and in all cases, 10 bins were used along each direction, for a total of 1000 bins.

Fig. 3.6 shows plots comparing the pixel value statistics of the epitome and those of the original image as a function of the size of the epitome. As expected, as the epitome becomes larger, the statistics of the epitome converges to the original image's distribution. For both images, gains in modelling the statistics of the image are initially quick as the epitome increases in size, but tapers off. Such a plot can be used to determine an appropriate size of the epitome by selecting the size that meets certain criteria, eg. models a sufficient amount of the statistics. Some threshold could be used to act as a knob to adjust the size of the epitome.

Similar plots for the horizontal and vertical derivatives are shown in Fig. 3.7, the

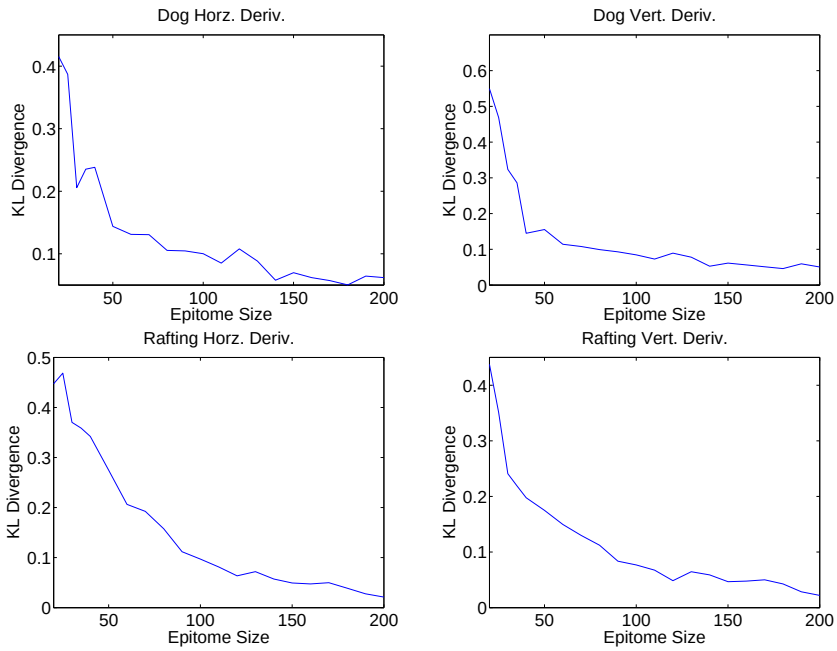

Figure 3.7: Image derivative comparison.

bandpass filter from Sec. 3.2.3 in Fig. 3.8, and the wavelet decomposition statistics in Fig. 3.9. The same pattern emerges that the small epitomes do not have the same statistics as the original, but the epitome is quickly able to model the statistics as the epitome size is increased, even when the epitome is significantly smaller than the original image. The epitome seems to be able to model all of the measured statistics well. In particular, the derivatives and bandpass filter response have distributions quite close to the original, which is not too surprising since both are very local measurements and the epitome excels at local features. The wavelet measure is like an amalgamation of both the derivative and bandpass result, but taken in a multi-resolution fashion. Correspondingly, the epitome also performs well with respect to the wavelet response, which shows that while the epitome features are local, the features are also quite large and do have many of the same trends as the original image.

There is no correct size of the epitome, as it depends on the desired use of the epitome. For compression, there is a balance between the amount of compression and the quality of compression. In denoising, the balance is between a large enough epitome to capture the

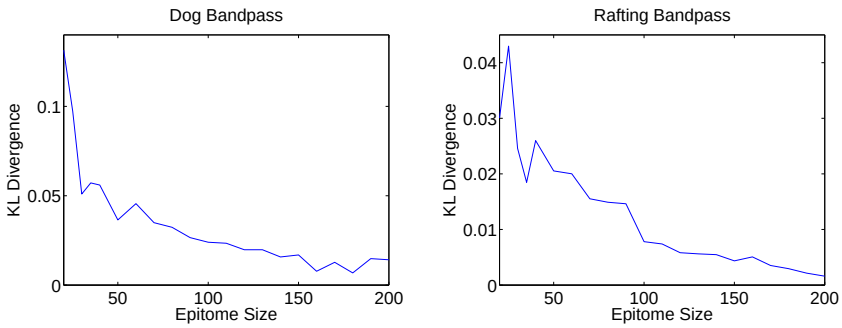

Figure 3.8: Bandpass filter output comparison.

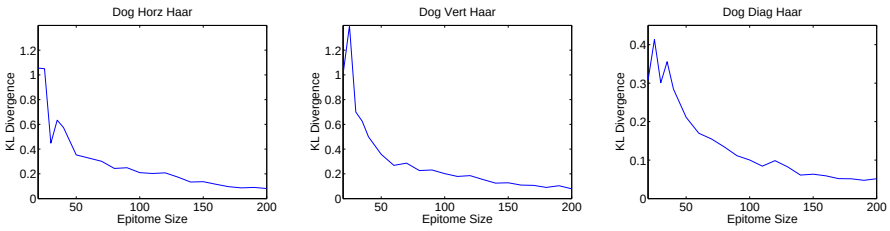

Figure 3.9: The horizontal, vertical, and diagonal wavelet filter response comparison.

desired features, but small enough to have the noise averaged out by using the epitome as a sort of regularizer. In some image processing applications, the trade-off is between the number of features in the epitome and computation time since a larger epitome requires more computations; however, in many cases, exact features are not needed, only representative ones. Further, the size of the epitome is dependent on the image itself, since an image with a lot of repeated patterns requires a smaller epitome than an image of the same size with no repeated patterns. Computing one or more of the statistics shown here provide a more systematic way to gauge the size of the epitome, rather than the current method of hand-waving and trial and error. For example, for the dog image, it would seem that a good balance between size and form is an epitome of size 100x100, as all the curves show that at this level, the epitome reflects the original statistics decently. It would also appear that the rafting image would need an epitome slightly larger, which makes sense since there are fewer repeated patterns than in the dog image.

While not particularly useful in its current form since nearly an exhaustive search of

epitome sizes was performed, this analysis could prove to be much more useful with a dynamically growing epitome. The epitome could be resized during learning, i.e. start with a small epitome and extend it along the edges, and the epitome can be made to stop growing once either particular statistics reach a particular threshold w.r.t. the original image or if the epitome image statistics are seen to plateau. Further, if it is seen that either the horizontal or vertical derivatives are performing poorly, then the epitome could be extended only by its width or height, breaking away from the square epitome sizes.

3.4 Conclusion

The epitome has been shown to be capable of modelling a number of different statistics of the original image while maintaining a far fewer number of pixels. The statistics were carefully chosen as the idea of the epitome had to be taken into account since the image epitome is not meant to be identical to the original image - in fact the epitome purposely changes the statistics of the image in order to represent its features in a smaller space. Plots of the KL-divergence between the image epitome statistics and the input image statistics as a function of the epitome size can act as a guide to determining the size of the epitome since in general, the epitome is desired to accurately model the features of the input image. This technique could prove to be very powerful when used in conjunction with a system where the epitome is resized during learning with the image statistics acting as a guide as to how to resize the epitome and when to stop.

Chapter 4

Efficiently Computing Epitomes

4.1 Introduction

A common operation in machine learning involves comparing patches in one image and those in a second. This situation arises in such problems as clustering, object and feature tracking [1], image epitome [25], texture synthesis [31], and stereo vision [20]. In these problems, it is desired to take a patch in one image and find the patch most closely resembling it in a second image. The metric for comparing patches varies, from Euclidean, Minkowski, and Mahalanobis distances to correlations. It may also be the case that this is needed to be repeated for every patch in the first image, where the patches are allowed to overlap; this is known as the nearest neighbour problem for patches. Further, it often arises in learning in these problems that what is needed is the actual distance between patches and not just the closest match. Complicating matters further, in some problems, it is beneficial to take into consideration patches of varying sizes and shapes. The difficulty is that this problem is very computationally intensive and in this chapter, a novel algorithm, the shifted cumulative sum algorithm is presented as an efficient solution to this patch comparison problem.

A variety of algorithms are known for solving this patch comparison problem. This

chapter does not consider approximate algorithms for solving the nearest neighbour problem [2], rather, it is restricted to algorithms that produce exact solutions. The naïve brute force algorithm is used in this chapter as a benchmark algorithm. A slight modification to the brute force algorithm allows the algorithm to break early for the nearest neighbour problem. The annulus bound algorithm attempts to use a bound to limit the number of patches that need to be considered as the nearest neighbour for a given patch. By manipulating the distance equation, the problem can be reduced to a number of convolutions, which can be efficiently performed with fast Fourier transforms. KD-trees [5] have had recent popularity, but tend to suffer in high dimensional problems such as those involving comparison of patches containing large number of pixels and are not considered in this chapter. Finally, an algorithm is proposed called the shifted cumulative sum algorithm that is invariant to the dimensionality of the problem (the patch size) and can simultaneously compare patches of any size and rectangular shape.

In Sec. 4.2, algorithms for solving the patch comparison problem are described. The shifted cumulative sum algorithm is presented as an efficient solution to this problem in Sec. 4.3. The results of experiments performed with these algorithms on the image epitome problem are then presented in Sec. 4.5.

4.2 Computing patch distances

In performing patch comparisons, it is often the case that squared Euclidean distances are used. Consider an NxN image, X, where $X^{(m)}$ is the m^{th} patch in the image of size PxP. If the image is taken to be circularly wrapped at the edges, then there are N^2 patches in X to be considered in the patch comparison problem. Similarly, for a second image, Y, of size KxK, there are K^2 patches, with $Y^{(n)}$ representing the n^{th} patch in Y. With this notation, the squared Euclidean distance between patch $X^{(m)}$ and $Y^{(n)}$ is

given by:

$$\|X^{(m)} - Y^{(n)}\|^2 = \sum_{i=1}^{P}\sum_{j=1}^{P}\left(X_{ij}^{(m)} - Y_{ij}^{(n)}\right)^2 \tag{4.1}$$

In the nearest neighbour problem, (4.1) must be minimized over n for a fixed m. When all distances between all patches are desired, (4.1) must be computed for all m and n.

In the epitome model described in Sec. 2.2, both learning and inference involve the computation of Euclidean distances between patches in the image and patches in the epitome because of an underlying normal distribution assumption. In learning, distances between all patches are required. Inference under this model involves maximizing (2.7) over all possible mappings, $\mathcal{T}_k$. This computation is equivalent to determining the nearest neighbour in the epitome for a given patch, $\boldsymbol{Z}_k$, but where a weighted Euclidean distance is used.

Many algorithms exist for solving the patch comparison problem presented here. This chapter investigates some of the most common algorithms, including, the brute force, brute force with early break, annulus bound, and an algorithm using the fast Fourier transform. The shifted cumulative sum algorithm is presented in the next section for efficiently computing distances in patch-based models. The algorithms presented in this chapter are applied to the epitome problem to evaluate their performance and the results are provided in the next section.

4.2.1 Brute force

The brute force algorithm is a direct implementation of (4.1) and for the nearest neighbour problem, it simply keeps track of the minimum patch distance for each $X^{(m)}$. The order of complexity for both patch comparison problems with this algorithm is $O(N^2K^2P^2)$ because the Euclidean distance must be computed between each patch in X and Y.

4.2.2 Brute force with early break

The brute force with early break algorithm is a modification of the brute force algorithm that solves the nearest neighbour problem. For nearest neighbour, only the minimum distance is of interest, thus at any point during the algorithm, if the current partial distance calculation is larger than the current minimum distance, then the current calculation can be stopped. As with the brute force algorithm, this algorithm has complexity $O(N^2K^2P^2)$.

4.2.3 Annulus bound

The annulus bound algorithm applies only to the nearest neighbour problem and is based on the triangle inequality [4]. This algorithm uses bounds to reduce the number of distance computations performed when seeking the nearest neighbour to a given point: Each quantity in (4.2) is shown graphically in Fig. 4.1. This equation states that if a

$$\begin{aligned} &\text{if } \|X-Z\| \leq \|X-Y\|, \text{ then} \\ &\big|\|X-R\| - \|X-Y\|\big| \leq \|Z-R\| \text{ and} \\ &\|Z-R\| \leq \|X-R\| + \|X-Y\| \end{aligned} \tag{4.2}$$

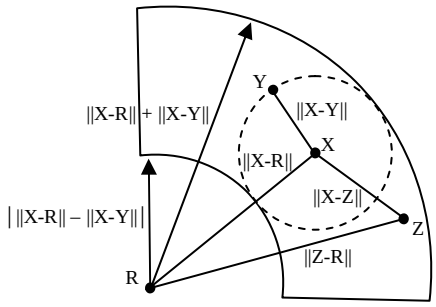

Figure 4.1: The annulus bound.

point Z is closer to X than Y, then it lies within some bounds. These bounds form an

annulus about the reference point, R. Typically, the reference point is chosen to be the origin as it minimizes the number of computations required. While this bound is not tight, depending on the data, it may eliminate many points out of consideration. As with the previous algorithms, this algorithm has complexity $O(N^2K^2P^2)$.

4.2.4 Fast Fourier transform

The fast Fourier transform (FFT) based algorithm arises from squaring out (4.1) to give:

$$\sum_{i=1}^{P}\sum_{j=1}^{P}\Big(\big(X_{ij}^{(m)}\big)^2 - 2X_{ij}^{(m)}Y_{ij}^{(n)} + \big(Y_{ij}^{(n)}\big)^2\Big) \tag{4.3}$$

Each of these terms in this expanded square can be computed efficiently for all patches using the FFT to perform convolutions. Each $X^{(m)}$ patch must be convolved with the Y image. Each convolution has a complexity of $O(K^2 \log K)$, so the overall complexity is $O(N^2K^2 \log K)$.

FFT

If using a L^2-norm or correlation distance measure, then the fast Fourier transform (FFT) can be used to efficiently compute the distance between a patch, $\mathbf{x}$, and all the other patches in an image or video sequence, $\mathbf{y}_i$.

$$D_{L^2}(\mathbf{x}, \mathbf{y}_i) = \sum_k (x_k - y_{ik})^2 = \sum_k x_k^2 + \sum_k y_{ik}^2 - 2\sum_k x_k y_{ik} \tag{4.4}$$

$$D_C(\mathbf{x}, \mathbf{y}_i) = \frac{\sum_k (x_k - \bar{\mathbf{x}})(y_{ik} - \bar{\mathbf{y}}_i)}{\sigma_{\mathbf{x}}\sigma_{\mathbf{y}_i}} \tag{4.5}$$

$$= \frac{\sum_k x_k y_{ik} - \bar{\mathbf{y}}_i \sum_k x_k - \bar{\mathbf{x}} \sum_k y_{ik} + K\bar{\mathbf{x}}\bar{\mathbf{y}}_i}{\sigma_{\mathbf{x}}\sigma_{\mathbf{y}_i}} \tag{4.6}$$

In both cases, all the terms can be pre-computed independent of which two patches are being compared (and done efficiently using cumulative sum matrices as shown in the following section), except this term

$$\sum_k x_k y_{ik} \tag{4.7}$$

Because the $\mathbf{y}_i$ patches actually come from an image, the other patch comparisons involving $\mathbf{x}$ can be considered as just shifting the patch to a different position relative to the image $\mathbf{Y}$. This term in (4.7) for all patches $\mathbf{y}_i$ is just a correlation, which can be represented as a convolution and then efficiently computing using FFTs.

$$\rho_{\mathbf{x},\mathbf{Y}}[k] = \mathbf{x}[-k] * \mathbf{Y}[k] = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}[-k])\mathcal{F}(\mathbf{Y}[k])) \tag{4.8}$$

$$= \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}[k])^*\mathcal{F}(\mathbf{Y}[k])) \tag{4.9}$$

where $*$ represents the convolution operator, $\mathcal{F}$ represents the (discrete, 2D) Fourier transform, $\mathcal{F}^{-1}$ inverse Fourier transform, and $\mathcal{F}(\cdot)^*$ the complex conjugate of the Fourier transform, and $\mathbf{x}$ has been zero-padded to the size of $\mathbf{Y}$. For the equivalent 1D computation, the index into the correlation matrix to obtain the proper value for (4.7) is given by the following, where P is the size of the patch:

$$\sum_k x_k y_{ik} = \rho_{\mathbf{x},\mathbf{Y}}[P + i] \tag{4.10}$$

If there are N^2 patches in $\mathbf{X}$, $\mathbf{Y}$ is of size $K \times K$, and the patches are of size $P \times P$, a brute force algorithm has a complexity of $\mathcal{O}(N^2K^2P^2)$, but using FFTs, this is reduced to $\mathcal{O}(N^2K^2 \log K)$. If however, the patches in $\mathbf{X}$ are sampled coarsely, and there are only T such patches, then the computation is reduced to $\mathcal{O}(TK^2 \log K)$.

4.3 The shifted cumulative sum algorithm

A new algorithm is proposed in this section to solve the patch comparison problem. The shifted cumulative sum (SCS) algorithm efficiently and exactly computes distances between patches in two images by exploiting the overlapping nature of the patches and reusing computations. This algorithm has a low order of complexity and is invariant to the dimensionality of the patches. Additionally, patches of any size and rectangular shape are simultaneously computed with this algorithm.

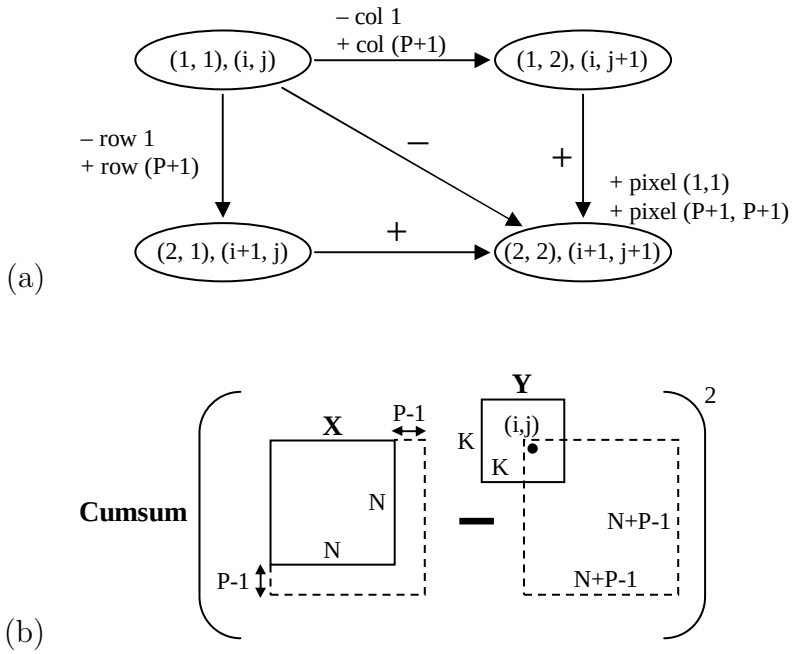

Figure 4.2: The shifted cumulative sum (SCS) algorithm. (a) The graph that motivated the SCS algorithm. (b) Graphical representation of the SCS algorithm.

The motivation for this algorithm is the graph shown in Fig. 4.2a. Each node in the graph represents the squared Euclidean distance between a patch in the image X and a patch in the second image Y. This graph shows that given the distance between patch (1,1) in the X image, $X^{(1,1)}$, and patch (i,j) in the Y image, $Y^{(i,j)}$, the distance between $X^{(1,2)}$ and $Y^{(i,j+1)}$ can be computed by subtracting the sum of the squared difference of the patches along the first column and adding those along the (P+1) column. The same can be done with the rows. With these three distances, a fourth distance can be obtained by a combination of these along with two extra squared difference elements. This graph can be extended for all patches in X where Y is circularly extended as required to give the distances $\|X^{(m,n)} - Y^{((m+i)\%K,(n+j)\%K)}\|$, $m, n \in 1 \ldots N$, where % is the modulus operator. This graph can be recreated for $i, j \in 1 \ldots K$ so that all the distances between patches in the two images are computed.

The direct implementation of this graph algorithm is inefficient, but it can be made

more efficient by considering the entire image and epitome at once rather than their constituent patches. This new algorithm is called the shifted cumulative sum (SCS) algorithm. Fig. 4.2b shows a graphical representation of the SCS algorithm. X and Y are the images from which patches are extracted. The dashed outlines indicate matrices obtained by circularly extending X and Y. Analogous to the graph in Fig. 4.2a, the epitome is shifted such that its (i,j) element aligns with the (1,1) element in X. Subtracting the two matrices of size $(N+P-1)\times(N+P-1)$ and performing element-wise squaring yields the squared difference of each element in X with the shifted Y. By forming the cumulative sum matrix, C, of this resulting matrix, patch distances can be computed as follows:

$$\|X^{(m,n)} - Y^{((m+i)\%K,(n+j)\%K)}\|^2 = C_{(m+P),(n+P)} - C_{m,(n+P)} - C_{(m+P),n} + C_{m,n} \quad (4.11)$$

As before, Y can be shifted in K^2 ways. The construction of each cumulative sum requires $O((N+P)^2) = O(N^2)$ operations since $N > P$, yielding an overall complexity of $O(N^2K^2)$. This bound is also the lower bound for the problem of computing all distances between the patches.

The SCS algorithm is not restricted to computing Euclidean distances, the more general Minkowski distance can be computed by using a different exponent upon the matrix prior to forming the cumulative sum matrix. Correlations can be performed by replacing the subtraction with element-wise multiplication and not performing the squaring operation. The Mahalanobis distance can also be computed with this algorithm provided that the covariance matrix is diagonal. This distance metric is essentially a weighted Euclidean distance and is exactly the situation encountered with the image epitome model. To compute this distance, element-wise division by the variance matrix is performed prior to computing the cumulative sum matrix.

The SCS algorithm is not without its faults. The algorithm requires a memory footprint on the order of magnitude as the largest image, in that a matrix of size $(N + P - 1)$x$(N + P - 1)$ is required for the cumulative sums. The main disadvan-

tage of this algorithm is that the patch comparisons are not computed in a typical order, so additional bookkeeping may be required.

4.4 Other considerations

4.4.1 Parallel and distributed computing

Many patch comparison problems can easily be parallelized because the comparison of one patch to a library of patches is independent of a second patch being compared to that same library of patches. With the epitome model, both inference and learning involves multiple patches from one image being compared with the patches in the epitome. The comparisons for each patch can be made into an independent operation that can be parallelized by using the batch update procedure, as opposed to the on-line one. In this case, the sufficient statistics are aggregated together before updating the epitome parameters.

The algorithms from Sec. 4.2 are trivially parallelized because each patch comparison against a library of patches is done separately and can be run in different threads or on different computers, with the sufficient statistics collected separately and combined together afterwards. The procedure is less clear with the SCS algorithm, but it nonetheless can still very much be parallelized.

With the SCS algorithm, partial statistics for the epitome update equations are collected with each shift of the cumulative sum matrices. The computational unit that can be separated is the computations for each shift, which can be run in different threads or on different computers. The difference here is that the partial statistics for all patch comparisons are computed with every shift, as opposed to the full patch comparison statistics for one patch. However, the partial statistics can just as easily be accumulated in different computational contexts and aggregated together afterwards.

4.4.2 Gaussian pyramids

With iterative techniques, a very popular method for computational speed-up is to use Gaussian pyramids. This method involves blurring and subsampling the image, potentially several times, and executing the algorithm involving patches on the smaller image. The intermediate result is used as the initialization for the higher resolution image, working down the pyramid back to the original resolution of the image. Often what is done is that the same patch size is used throughout the entire process, which effectively uses a larger patch size with reduced computational cost in the higher levels of the pyramid.

This technique can be used in conjunction with all the algorithms in this chapter and the computational gain is obtained by reducing the size of N and K for some iterations of the algorithm.

4.4.3 Reducing the number of patch comparisons

Another technique for reducing computational complexity of patch comparisons is to simply perform fewer patch comparisons. In the epitome model, not every possible patch in the image is required to be used to update the epitome since overlapping patches contain many of the same pixel values. This has an effect on the epitome model in terms of what features it learns, but depending on the application, this is acceptable. The patches extracted from the image can be done in several different ways, such as sampling on a regular grid with varying sampling frequency or a random sampling. It has been found that in order for the epitome model to properly learn features in an image, it is important for the patches to overlap so that boundary effects of the patches get averaged out.

The algorithms in Sec. 4.2 directly reduce their computational cost by the reduction in the number of patch computations because each operation works on each patch separately and reducing the computational complexity by reducing the N^2 operations. For example,

if the patches are sampled on a regular grid every $P/2$ pixels, i.e. the patches overlap by 50%, then the number of patch comparisons required are reduced to $O(N^2/P^2)$.

The SCS algorithm does not gain a significant speed-up with such patch sampling because regardless of the number of patches used, operations on the entire $O(N^2)$ image are necessary for all $O(K^2)$ shifts. The biggest advantage of the SCS algorithm is in sharing computations across all patch comparisons, even for different patch sizes. Effectively, with the computational complexity of $O(N^2K^2)$, you then get all patch comparisons of all patch sizes for free.

4.5 Experiments

All of the algorithms in Sec. 4.2 are able to solve the inference problem. Further, those that also solve the learning problem, namely, the brute force, FFT, and SCS algorithms, require the same number of computations to perform inference as it does to perform the E step in learning, assuming that inference is done for all patches used in learning. Thus, all the experiments for this chapter had the algorithms solve the inference problem. A number of these experiments were performed to evaluate the various algorithms under changes in problem parameters and the image data. In Sec. 4.5.1, a random image and epitome were used to evaluate the algorithms on such data, that is, the epitome was not representative of the image, which is the case in the early stages of learning. This type of data is the worst case for some of the algorithms. In Sec. 4.5.2, a learned epitome is used to evaluate the algorithms under conditions where patches in the image and epitome are similar.

4.5.1 Random data experiments

In these experiments, both the image and epitome were randomly generated from uniform distributions so the algorithms could be evaluated in the situation where the epitome does

not resemble the image, as in the early stages of learning. Experiments were performed to compare the algorithms with respect to changes in the image, epitome, and patch size. An experiment was also done using multiple patch sizes simultaneously

Three separate experiments were performed, each one separately testing the effect of the image, epitome, and patch size on the runtime of the algorithms. Plots of the results are shown in Fig. 4.3. Experiments were repeated ten times and were averaged to produced these plots. The small error bars associated with these plots are not shown for clarity. Each of the algorithms scaled quadratically with the size of the image as expected from the complexity analysis. All the algorithms except the FFT algorithm also scaled quadratically with the size of the epitome. The FFT algorithm scaled slightly greater as its complexity is given by $O(N^2K^2 \log K)$, where K is the size of the epitome. The algorithms also behaved as expected under changes to the patch size. The FFT and SCS algorithms were essentially invariant to the patch size, while the other algorithms scaled quadratically.

The SCS algorithm proved to be consistently faster than the other algorithms investigated in this chapter - generally faster by at least an order of magnitude. The algorithm scales well and has a low overhead. The FFT algorithm has a low order of complexity, but has such high overhead that it should not be considered unless the patch size is very large. The remaining algorithms all performed nearly the same. It is not surprising that the brute force algorithm was at times, faster than the other two, since they have some overhead and uniform random images and epitomes do not give much opportunity to save computations since all the patches are very similar.

In the image epitome, it is important that varying patch sizes be used in learning [25]. Using multiple patch sizes and shapes are important in other problems since image features are generally not square and of the same size. In performing patch comparisons, taking into consideration multiple patch sizes, some of the algorithms can reuse computations. Table 4.1 compares the results of simply running the algorithms once for

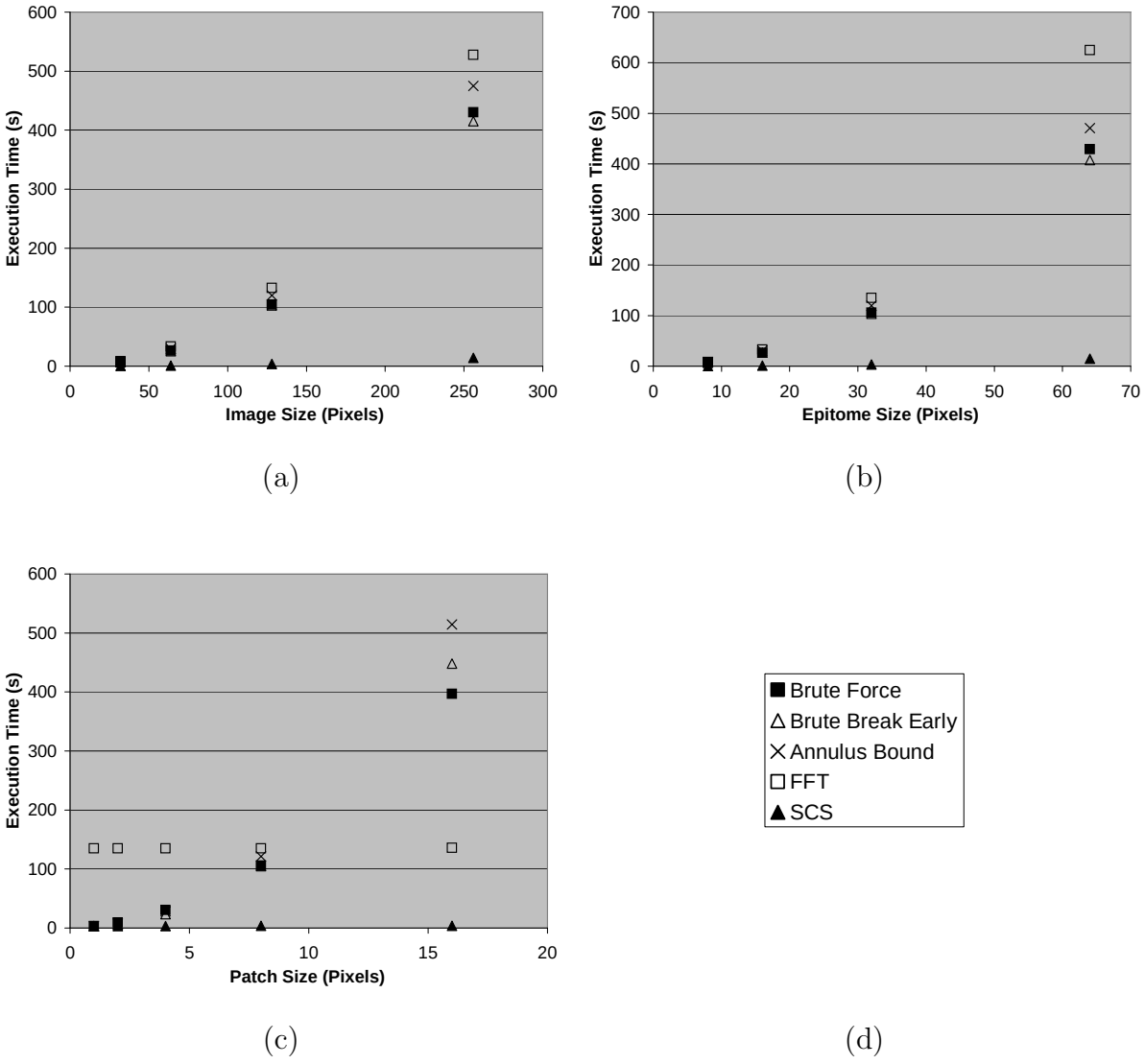

Figure 4.3: Plots showing the effect of (a) the image size, (b) the epitome size, and (c) the patch size on the execution times of the various algorithms (d) using uniform random images and epitomes. The base parameters are a 128x128 image, a 32x32 epitome, and 8x8 patches. In each of these plots, one of the parameters are varied while the others are held constant at the base values. These experiments were performed using Java with the Excelsior JET native compiler (obtained from http://www.excelsior-usa.com/) on a Pentium IV 2 GHz computer.

each patch size in Fig. 4.3c with the execution time of running the algorithms on the multiple patch sizes simultaneously. Since the brute force and break early algorithms did not reuse any computations, when multiple patch sizes were considered, their execution times did not change over simply running them multiple times. However, the annulus bound, FFT, and SCS algorithms improved as some computations were reused. The SCS algorithm improved the most and performed the best since in a single iteration, all patch distances for all patch sizes and shapes are computed simultaneously. The SCS algorithm incurs very little overhead in incorporating varying patch sizes and shapes as it has an order of complexity given by $O(N^2K^2 + MK^2)$, where M is the total number of patches taken from the image. Conversely, in the other algorithms, very few computations can be reused when multiple patch sizes and shapes are considered and essentially the algorithms must be repeated for each change in patch size.

Table 4.1: Experiment results with multiple patch sizes. The "Sum" column is the sum of the execution times from Fig. 4.3c where the patch size was varied. The "Multiple" column is the execution time of the algorithms when these patch sizes are considered simultaneously.

	Execution Time(s)	
Algorithm	**Sum**	**Multiple**
Brute Force	546.36	544.75
Brute Break Early	584.81	584.95
Annulus Bound	674.17	585.98
FFT	675.78	643.52
SCS	17.70	6.35

4.5.2 Learned data experiments

The annulus and break early algorithms are stochastic and are sensitive to the particular patches used. They are most effective when the image patches have a lot of variance and the epitome is representative of the image. To assess this, a real data experiment was performed.

One experiment was done using the 128x128 image and 32x32 epitome from Fig. 2.1 with 8x8 patches and a second used multiple patch sizes. The results of these experiments are shown in Table 4.2. In both experiments, the brute force, FFT, and SCS runtimes remained the same, but the annulus and break early algorithms executed significantly faster.

Table 4.2: Experiment results using a learned epitome of an image with patches of size 8x8 and square patches of size 1, 2, 4, 8, and 16

	Execution Time(s)	
Algorithm	**8x8**	**Multiple**
Brute Force	103.75	545.73
Brute Break Early	31.81	204.00
Annulus Bound	43.41	203.69
FFT	131.86	641.38
SCS	3.69	6.37

4.6 Conclusion

A novel algorithm, the shifted cumulative sum algorithm was proposed to efficiently and exactly compute distances in patch-based models and solve the nearest neighbour patch problem. This algorithm scales well and has a low overhead. In every experiment

performed in this chapter, the shifted cumulative sum algorithm executed faster than the other algorithms by at least an order of magnitude. The performance is even more pronounced when multiple patch sizes and shapes are used for patch comparisons. With the shifted cumulative sum algorithm, distances for all patch sizes and rectangular shapes are computed simultaneously in a manner that is invariant to the size of the patches. The main disadvantage of this algorithm is that the patch comparisons are not computed in a typical order, but this generally poses a minor annoyance rather than a problem.

Chapter 5

Video Epitome

5.1 Introduction

In this chapter, we address two problems: (1) How 2D image epitomes can be extended through time to form 3D space-time epitomes; (2) How epitomes can be compiled and applied in a computationally efficient manner and in particular, in a way that is significantly more efficient than using a library of patches. The extension to 3D introduces several issues due to the trade-off of space vs time and the computational issues introduced because of the great increase in the number of patches from the added time dimension. After describing a general framework for learning video epitomes, an extension to the epitome model is introduced to account for missing data, which allows the model to be used for a number of disparate applications.

5.2 Video epitomes

Fig. 5.1 outlines the procedure used to learn a video epitome. Viewing the input video as a 3D space-time volume, a large number of 3D training patches are drawn from the video. The learning algorithm is used to compile these patches into an "epitome" – a video that is smaller in both space and time, but contains many of the spatial and temporal patterns

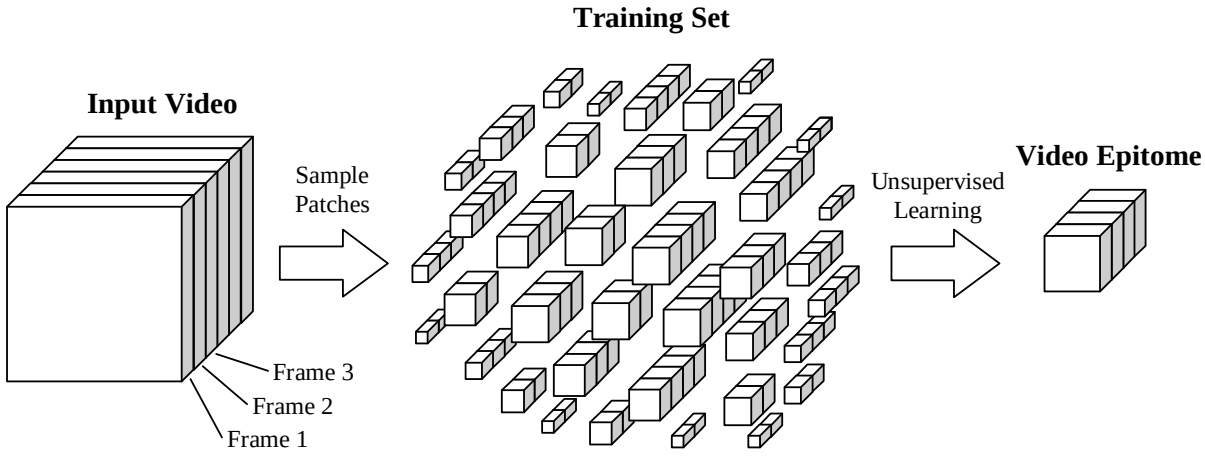

Figure 5.1: Learning the epitome of a video.

in the original input video. We derive the epitome learning algorithm by specifying a generative model, which explains how the input video can be generated from the epitome (in the opposite direction shown in the figure). The advantage of specifying a generative model is that the model is more adaptive than a non-generative technique and it can be used as a sub-component in other systems. Here, we introduce a 3D model of video similar to the 2D epitome model described in [25].

We treat a video sequence as a 3D array $\mathbf{v}_{x,y,t}$ of real-valued pixel measurements (R, G, and B color channels in this chapter), with $x \in: \{1, ..., X_v\}$, $y \in: \{1, ..., Y_v\}$, $t \in: \{1, ..., T_v\}$. The epitome e models the video using a set of probability distributions arranged on a grid of size $X_e \times Y_e \times T_e$, considerably smaller then the video, *i.e.*, $X_e Y_e T_e \ll X_v Y_v T_v$. We view the epitome $e_{x,y,t}$ as a 3D array of probability distributions. A particular pixel value $\mathbf{v}$ can be evaluated under any of the probability distributions in e. For example, for the epitome coordinates x_e, y_e, t_e , the probability density at the pixel value stored in the entry x_v, y_v, t_v of the video is $e_{x_e,y_e,t_e}(\mathbf{v}_{x_v,y_v,t_v})$. Since the pixel measurements are continuous in nature, it is necessary to parameterize each of the epitome distributions. In our experiments, we use a single parametric form, a three-dimensional Gaussian distribution parameterized by a different mean and diagonal covariance matrix

for each entry,

$$e_{x,y,t}(\,\cdot\,) = \mathcal{N}(\,\cdot\,; \mu_{x,y,t}, \phi_{x,y,t}), \tag{5.1}$$

where $\mu_{x,y,t}$ is the mean and $\phi_{x,y,t}$ is the covariance matrix (*e.g.* for RGB values). The diagonal covariance matrix decouples color channel computations. Because of this, we will treat the measurements $\mathbf{v}_{x,y,t}$ as scalar in the following derivations, which the reader can use to derive the full color model.

The epitome models the video by modeling 3D patches sampled from the video. These patches can have any shape, but to keep notation simple, we will assume each patch has linear, axis-aligned boundaries, and we think of each patch as a "cube". Each patch is defined in terms of the ordered set of pixel coordinates in the patch, $\mathcal{S}$. For example, a $10 \times 10 \times 5$ video patch starting at position $8, 9$ in frame 7 of the video is described by the set $\mathcal{S} = \{8, .., 17\} \times \{9, .., 18\} \times \{7, .., 11\}$. We assume the coordinates in $\mathcal{S}$ are ordered, so that $\mathcal{S}(k)$ refers to the kth coordinate in $\mathcal{S}$, *e.g.*, $\mathcal{S}(1) = (8, 9, 7)$ in the above example.

While $\mathbf{v}$ denotes the observed pixel values at all coordinates in the video, $\mathbf{v}_\mathcal{S}$ denotes the observed pixel values in a small video cube at coordinates $\mathcal{S}$, and $\mathbf{c}_\mathcal{S}$ denotes the pixel values in the same cube, as predicted by the epitome. As described below, the goal of the learning algorithm is to make the predicted video cubes similar to the observed video cubes, *i.e.*, $\mathbf{c}_\mathcal{S} \approx \mathbf{v}_\mathcal{S}$. The cube $\mathbf{c}_\mathcal{S}$ is modeled using a set of distributions in the epitome. We use $e_\mathcal{T}$ to denote the set of distributions from the epitome e at coordinates $\mathcal{T}$. Assuming that the cube corresponding to $\mathcal{T}$ is equal in size to cube corresponding to $\mathcal{S}$, we can define a one-to-one coordinate correspondence so that the set ordering is preserved, and the notation $\mathcal{T}_\mathcal{S}$ is used. The probability density evaluated using distributions at coordinates $\mathcal{T}$ and predicted values $\mathbf{c}_\mathcal{S}$ is then $e_\mathcal{T}(\mathbf{c}_\mathcal{S})$:

$$p(\mathbf{c}_\mathcal{S}|\mathcal{T}_\mathcal{S}) = e_{\mathcal{T}_\mathcal{S}}(\mathbf{c}_\mathcal{S}) = \prod_{k=1}^{|\mathcal{T}|} e_{\mathcal{T}_\mathcal{S}(k)}(\mathbf{c}_{\mathcal{S}(k)}). \tag{5.2}$$

The above equation assumes independence of pixels given the responsible set of epitome

distributions, with the idea of capturing correlations simply through mapping sets of measurements to the overlapping sets of distributions.

The above equation describes the probability model for an individual cube. To obtain a probability model for the entire input video, $\mathbf{v}$, we need to address the issue of how to account for overlapping cubes. If two sets of coordinates $\mathcal{S}$ and $\mathcal{S}'$ overlap, then the corresponding cubes $\mathbf{c}_{\mathcal{S}}$ and $\mathbf{c}_{\mathcal{S}'}$ are used to predict overlapping cubes in the input video $\mathbf{v}$, and so they should make similar predictions for overlapping pixels. The most obvious approach to accounting for overlap is to include a constraint that the predictions are identical in regions of overlap. However, this approach requires the introduction of a partition function, making learning significantly more difficult. We take a different, novel, approach [25], where we treat the cubes $\mathbf{c}_{\mathcal{S}}$ as independent even if they share coordinates, but enforce the agreement in the overlapping regions during inference, as described in the next section.

We now define a generative model of video sequences that is suitable for all applications described in the experimental section. The first step in the generative process consists of generating a predicted cube $\mathbf{c}_{\mathcal{S}}$ for every possible coordinate set $\mathcal{S}$ in the input video. This is accomplished by first randomly choosing a patch $\mathcal{T}_{\mathcal{S}}$ from the epitome using a uniform distribution, and then generating $\mathbf{c}_{\mathcal{S}}$ using the distribution $e_{\mathcal{T}_{\mathcal{S}}}(\mathbf{c}_{\mathcal{S}})$ defined by (5.2). Then, for each pixel coordinate x, y, t in the video, all overlapping cubes $\{\mathcal{S} : (x, y, t) \in \mathcal{S}\}$ are combined to make a single prediction for the observed pixel $\mathbf{v}_{x,y,t}$ at that coordinate. This generative process is illustrated in Fig. 5.2.

During free energy minimization, we will force the predictions to agree, so the exact form of the combination function is not important. Here, we assume that to generate a consistent video pixel $\mathbf{v}_{x,y,t}$, the contributions from all overlapping video cubes $\{\mathcal{S} : (x, y, t) \in \mathcal{S}\}$ are averaged, and Gaussian noise with variance $\sigma^2_{x,y,t}$ is added to all three

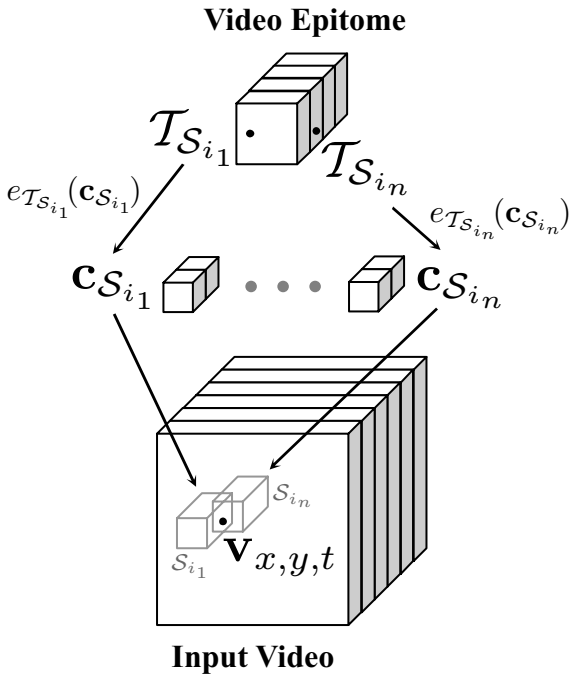

Figure 5.2: Generative process for a single video pixel, $\mathbf{v}_{x,y,t}$, which is represented by a dot in the input video. The overlapping video cubes in the input video containing the pixel x, y, t are $\{\mathcal{S}_{i_1}, \ldots, \mathcal{S}_{i_n}\}$, outlined in gray in the input video. For each of these video cubes, there is a predicted cube. These predicted cubes, $\{\mathbf{c}_{\mathcal{S}_{i_1}}, \ldots, \mathbf{c}_{\mathcal{S}_{i_n}}\}$, are generated from the epitome by randomly selecting locations in the epitome $\{\mathcal{T}_{\mathcal{S}_{i_1}}, \ldots, \mathcal{T}_{\mathcal{S}_{i_n}}\}$, represented by dots in the video epitome, and then generating the pixel values for the predicted cubes under the distribution given by the epitome for these locations, $e_{\mathcal{T}_{\mathcal{S}_{i_1}}}(\mathbf{c}_{\mathcal{S}_{i_1}}), \ldots, e_{\mathcal{T}_{\mathcal{S}_{i_n}}}(\mathbf{c}_{\mathcal{S}_{i_n}})$. Each of these cubes $\mathbf{c}_{\mathcal{S}_i}$, make predictions for the pixel $\mathbf{v}_{x,y,t}$ and these predictions are combined to generate a single pixel value.

channels:

$$p(\mathbf{v}_{x,y,t}|\{\mathbf{c}_{\mathcal{S}} : (x,y,t) \in \mathcal{S}\}) = \mathcal{N}(\mathbf{v}_{x,y,t}; \frac{\sum_{\mathcal{S}} \sum_k [\mathcal{S}(k) = (x,y,t)] \mathbf{c}_{\mathcal{S},k}}{\sum_{\mathcal{S}} \sum_k [\mathcal{S}(k) = (x,y,t)]}, \sigma^2_{x,y,t})$$

$$p(\mathbf{v}|\{\mathbf{c}_{\mathcal{S}}\}) = \prod_{x,y,t} p(\mathbf{v}_{x,y,t}|\{\mathbf{c}_{\mathcal{S}} : (x,y,t) \in \mathcal{S}\}), \tag{5.3}$$

where [ ] is Iverson's indicator function, *i.e.*, $[true] = 1$, $[false] = 0$. We use the notation $\mathbf{c}_{\mathcal{S},k}$ for the k-th pixel in $\mathbf{c}_{\mathcal{S}}$ to emphasize that the video cubes $\mathbf{c}_{\mathcal{S}}$ are treated

as independent, and so a pixel in the video cube $\mathbf{c}_\mathcal{S}$ is not uniquely defined by the global coordinate $\mathcal{S}(k)$, and is instead, potentially different from the pixels in other cubes that overlap $\mathcal{S}(k)$. However, as described by the above equations, for each coordinate (x, y, t), a single final pixel $\mathbf{v}_{x,y,t}$ is generated by adding noise to the average of all video cubes overlapping with coordinate (x, y, t).

Usually, a subset of the input video patches provides accurate sufficient statistics for learning an epitome. In practice, we randomly sample the set $\{\mathcal{S}\}$ of video cubes to be used for learning (*e.g.* from the set of all video cubes of a certain size) so that with high probability, or absolute certainty (*e.g.* with constrained random sampling), we would expect to see every pixel from the input video in at least one video cube. The joint distribution over all variables can be written $p(\mathbf{v}, \{\mathbf{c}_\mathcal{S}, \mathcal{T}_\mathcal{S}\}) = p(\mathbf{v}|\{\mathbf{c}_\mathcal{S}\}) \prod_\mathcal{S} p(\mathbf{c}_\mathcal{S}|\mathcal{T}_\mathcal{S})p(\mathcal{T}_\mathcal{S})$. We often assume $p(\mathcal{T}_\mathcal{S})$ is uniform for the sake of simplicity.

5.3 Learning video epitomes

Learning under this model using the expectation-maximization algorithm is not possible because the exact posterior is intractable. To overcome this problem, a variational technique is used to approximate the posterior. Learning can be viewed as minimizing a free energy cost function [18]:

$$F = \sum_\mathcal{S} \sum_{\mathcal{T}_\mathcal{S}} \int_{\mathbf{c}_\mathcal{S}} q(\{\mathcal{T}_\mathcal{S}, \mathbf{c}_\mathcal{S}\}) \log \frac{q(\{\mathcal{T}_\mathcal{S}, \mathbf{c}_\mathcal{S}\})}{p(\mathbf{v}, \{\mathbf{c}_\mathcal{S}, \mathcal{T}_\mathcal{S}\})}, \tag{5.4}$$

where $q(\{\mathcal{T}_\mathcal{S}, \mathbf{c}_\mathcal{S}\})$ is the auxiliary joint probability distribution over the set of epitome patches $\{\mathcal{T}_\mathcal{S}\}$ and the set of video cubes $\{\mathbf{c}_\mathcal{S}\}$ for all coordinate patches $\{\mathcal{S}\}$. It turns out that the free energy [32] bounds the log-likelihood of the input video, $F \geq -\log p(\mathbf{v})$, where $p(\mathbf{v}) = \sum_{\{\mathcal{T}_\mathcal{S}\}} \int_{\{\mathbf{c}_\mathcal{S}\}} p(\mathbf{v}, \{\mathbf{c}_\mathcal{S}, \mathcal{T}_\mathcal{S}\})$, the proof of which can be found in Sec. 5.4. So, by optimizing F, we can maximize $p(\mathbf{v})$. In fact, the closer q is to the true posterior $p(\{\mathcal{T}_\mathcal{S}, \mathbf{c}_\mathcal{S}\}|\mathbf{v})$, the tighter the bound, so q is an approximation to the posterior distribution. By choosing an appropriate form for q, we can achieve two goals: Obtain an efficient

inference and learning algorithm by decoupling variables [32]; Constrain inference and learning to the space where overlapping cubes agree, as proposed in [25]. We choose the q-distribution as follows:

$$q(\{\mathcal{T}_\mathcal{S}, \mathbf{c}_\mathcal{S}\}) = \prod_\mathcal{S} q(\mathcal{T}_\mathcal{S}) q(\mathbf{c}_\mathcal{S}). \tag{5.5}$$

$q(\mathcal{T}_\mathcal{S})$ is a discrete distribution over all possible patch locations in the epitome shaped as $\mathcal{S}$. To enforce the overlap constraint, $q(\mathbf{c}_\mathcal{S})$ is a product of Dirac functions:

$$q(\mathbf{c}_\mathcal{S}) = \prod_k \delta(\mathbf{c}_{\mathcal{S},k} - \nu_{\mathcal{S}(k)}). \tag{5.6}$$

As before, $\mathbf{c}_{\mathcal{S},k}$ denotes the pixel $\mathcal{S}(k)$ in cube $\mathbf{c}_\mathcal{S}$. $\nu_{\mathcal{S}(k)}$ denotes the variational parameter for the pixel $\mathcal{S}(k) = (x, y, t)$, which is *shared* by all patches S that contain the coordinate (x, y, t), thus constraining the patches to agree in overlapping pixels.

The free energy obtained using the above q-distribution leads to a tractable iterative learning algorithm. Setting to zero the derivatives of F w.r.t. the posterior cube colors $\nu_{x,y,t}$, the details of which are in Sec. 5.4, we obtain the following update rule:

$$\nu_{x,y,t} \leftarrow \frac{\frac{\mathbf{v}_{x,y,t}}{\sigma^2_{x,y,t}} + \sum_{\mathcal{S},k:\mathcal{S}(k)=(x,y,t)} \sum_{\mathcal{T}_\mathcal{S}} q(\mathcal{T}_\mathcal{S}) \frac{\mu_{\mathcal{T}_\mathcal{S}(k)}}{\phi_{\mathcal{T}_\mathcal{S}(k)}}}{\frac{1}{\sigma^2_{x,y,t}} + \sum_{\mathcal{S},k:\mathcal{S}(k)=(x,y,t)} \sum_{\mathcal{T}_\mathcal{S}} q(\mathcal{T}_\mathcal{S}) \frac{1}{\phi_{\mathcal{T}_\mathcal{S}(k)}}}. \tag{5.7}$$

This update sets the "hidden video" ν to a weighted combination of the input video and the top-down prediction for the video, as given by the current epitome. The weights are the inverse noise variances for the video and the epitome. Setting to zero the derivatives of F w.r.t. the posterior epitome responsibilities $q(\mathcal{T}_\mathcal{S})$, we obtain

$$q(\mathcal{T}_\mathcal{S}) \leftarrow \frac{p(\mathcal{T}_\mathcal{S}) e_{\mathcal{T}_\mathcal{S}}(\nu_\mathcal{S})}{\sum_\mathcal{T} p(\mathcal{T}) e_\mathcal{T}(\nu_\mathcal{S})}. \tag{5.8}$$

This update is similar to computing the responsibilities of components in a mixture of Gaussians. For each cube $\nu_\mathcal{S}$, the distribution over its position in the epitome is proportional to the probability it was generated from each position. To perform inference on video $\mathbf{v}$ when the epitome is given, these two updates can be iterated until convergence.

If the epitome is to be learned from the video $\mathbf{v}$, then the two equations above are iterated in combination with the following two updates, obtained by setting to zero the derivatives of F w.r.t. the epitome parameters $\mu_{x,y,t}$ and $\phi_{x,y,t}$:

$$\mu_{x_e,y_e,t_e} \leftarrow \frac{\sum_{\mathcal{T},k:\mathcal{T}(k)=(x_e,y_e,t_e)} \sum_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}}=\mathcal{T})\nu_{\mathcal{S}(k)}}{\sum_{\mathcal{T},k:\mathcal{T}(k)=(x_e,y_e,t_e)} \sum_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}}=\mathcal{T})} \tag{5.9}$$

$$\phi_{x_e,y_e,t_e} \leftarrow \frac{\sum_{\mathcal{T},k:\mathcal{T}(k)=(x_e,y_e,t_e)} \sum_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}}=\mathcal{T})(\nu_{\mathcal{S}(k)}-\mu_{x_e,y_e,t_e})^2}{\sum_{\mathcal{T},k:\mathcal{T}(k)=(x_e,y_e,t_e)} \sum_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}}=\mathcal{T})}. \tag{5.10}$$

The first update sets the mean pixel value in the epitome to the average value of all pixel values from video cubes $\nu_{\mathcal{S}}$, weighted by the probability that the cube is aligned with the pixel in the epitome, $q(\mathcal{T}_{\mathcal{S}}=\mathcal{T})$. The second update is similar, except it accounts for the variance, not the mean value. All inference algorithms in this chapter are based on these four equations and the derivations for these equations are shown in Sec. 5.4. In Sec. 5.5, we address the issue of the relative dimensions of the epitome, Sec. 5.6 explains how the above equations define different video processing applications, and in Sec. 5.7 we derive an efficient algorithm that implements these equations.

5.4 Derivation of the video epitome learning algorithm

Learning under the video epitome model is performed by using a variational expectation-maximization algorithm, which centers around minimizing the free energy cost function in (5.4). The free energy bounds the log-likelihood of the input video, so by optimizing F, the log-likelihood of the input video is maximized, the proof of which can be shown

by using Jensen's inequality:

$$
\begin{aligned}
F &= \sum_{\mathcal{S}} \sum_{\mathcal{T}_{\mathcal{S}}} \int_{\mathbf{c}_{\mathcal{S}}} q(\{\mathcal{T}_{\mathcal{S}}, \mathbf{c}_{\mathcal{S}}\}) \log \frac{q(\{\mathcal{T}_{\mathcal{S}}, \mathbf{c}_{\mathcal{S}}\})}{p(\mathbf{v}, \{\mathbf{c}_{\mathcal{S}}, \mathcal{T}_{\mathcal{S}}\})} \\
&= -\sum_{\mathcal{S}} \sum_{\mathcal{T}_{\mathcal{S}}} \int_{\mathbf{c}_{\mathcal{S}}} q(\{\mathcal{T}_{\mathcal{S}}, \mathbf{c}_{\mathcal{S}}\}) \log \frac{p(\mathbf{v}, \{\mathbf{c}_{\mathcal{S}}, \mathcal{T}_{\mathcal{S}}\})}{q(\{\mathcal{T}_{\mathcal{S}}, \mathbf{c}_{\mathcal{S}}\})} \\
&\geq -\log \sum_{\mathcal{S}} \sum_{\mathcal{T}_{\mathcal{S}}} \int_{\mathbf{c}_{\mathcal{S}}} q(\{\mathcal{T}_{\mathcal{S}}, \mathbf{c}_{\mathcal{S}}\}) \frac{p(\mathbf{v}, \{\mathbf{c}_{\mathcal{S}}, \mathcal{T}_{\mathcal{S}}\})}{q(\{\mathcal{T}_{\mathcal{S}}, \mathbf{c}_{\mathcal{S}}\})} \\
&= -\log \sum_{\mathcal{S}} \sum_{\mathcal{T}_{\mathcal{S}}} \int_{\mathbf{c}_{\mathcal{S}}} p(\mathbf{v}, \{\mathbf{c}_{\mathcal{S}}, \mathcal{T}_{\mathcal{S}}\}) \\
&= -\log p(\mathbf{v}).
\end{aligned}
$$

Learning progresses by iteratively updating the approximation to the true posterior, $q(\{\mathcal{T}_{\mathcal{S}}, \mathbf{c}_{\mathcal{S}}\})$ and the variational parameters: the hidden video, ν, and the mean and variance of the epitome, μ and ϕ. The update equations for the q-distribution and the parameters (5.7 -5.10) are obtained by setting to zero the derivatives of the free energy cost function, the procedure of which is shown below.

The posterior epitome responsibilities $q(\mathcal{T}_{\mathcal{S}})$ are obtained by using a Lagrange multi-

plier during optimization to constrain q to be a proper distribution:

$$\begin{aligned}
\frac{\partial\Big(\lambda(\sum_{\mathcal{T}} q(\mathcal{T})-1)+F\Big)}{\partial q(\mathcal{T}_{\mathcal{S}})} &= \frac{\partial}{\partial q(\mathcal{T}_{\mathcal{S}})}\Bigg(\lambda\Big(\sum_{\mathcal{T}} q(\mathcal{T})-1\Big) \\
&\quad + \sum_{\mathcal{S}}\sum_{\mathcal{T}}\int_{\mathbf{c}_{\mathcal{S}}}\Big(\prod_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}})q(\mathbf{c}_{\mathcal{S}})\Big)\log\frac{\prod_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}})q(\mathbf{c}_{\mathcal{S}})}{p(\mathbf{v},\{\mathbf{c}_{\mathcal{S}},\mathcal{T}_{\mathcal{S}}\})}\Bigg) \\
0 &= \frac{\partial}{\partial q(\mathcal{T}_{\mathcal{S}})}\Big(\lambda q(\mathcal{T}_{\mathcal{S}}) + q(\mathcal{T}_{\mathcal{S}})\log q(\mathcal{T}_{\mathcal{S}}) \\
&\quad - q(\mathcal{T}_{\mathcal{S}})\int_{\mathbf{c}_{\mathcal{S}}}\Big(\prod_{\mathcal{S}} q(\mathbf{c}_{\mathcal{S}})\Big)\log p(\mathbf{v},\{\mathbf{c}_{\mathcal{S}}\},\mathcal{T}_{\mathcal{S}})\Big) \\
0 &= \lambda + \log q(\mathcal{T}_{\mathcal{S}}) + 1 - \log p(\mathbf{v},\{\nu_{\mathcal{S}}\},\mathcal{T}_{\mathcal{S}})
\end{aligned}$$

$$\begin{aligned}
q(\mathcal{T}_{\mathcal{S}}) &= \exp(-\lambda-1)p(\mathbf{v},\{\nu_{\mathcal{S}}\},\mathcal{T}_{\mathcal{S}}) \\
q(\mathcal{T}_{\mathcal{S}}) &= \exp(-\lambda-1)p(\mathbf{v}|\{\nu_{\mathcal{S}}\})p(\mathcal{T}_{\mathcal{S}})e_{\mathcal{T}_{\mathcal{S}}}(\nu_{\mathcal{S}}) \\
1 = \sum_{\mathcal{T}} q(\mathcal{T}) &= \exp(-\lambda-1)p(\mathbf{v}|\{\nu_{\mathcal{S}}\})\sum_{\mathcal{T}} p(\mathcal{T})e_{\mathcal{T}}(\nu_{\mathcal{S}}) \\
\exp(-\lambda-1)p(\mathbf{v}|\{\nu_{\mathcal{S}}\}) &= \frac{1}{\sum_{\mathcal{T}} p(\mathcal{T})e_{\mathcal{T}}(\nu_{\mathcal{S}})},
\end{aligned}$$

by substituting this last result into the equation for $q(\mathcal{T}_{\mathcal{S}})$ above, the following update is obtained:

$$q(\mathcal{T}_{\mathcal{S}}) \leftarrow \frac{p(\mathcal{T}_{\mathcal{S}})e_{\mathcal{T}_{\mathcal{S}}}(\nu_{\mathcal{S}})}{\sum_{\mathcal{T}} p(\mathcal{T})e_{\mathcal{T}}(\nu_{\mathcal{S}})}.$$

The update equation for the posterior cube colors ν_i, where $i=(x,y,t)$, are derived by minimizing the free energy w.r.t. ν_i, and as such, only terms involving ν_i need to be

considered,

$$\begin{aligned}
\frac{\partial F}{\partial \nu_i} &= -\frac{\partial}{\partial \nu_i} \sum_{\mathcal{S}} \sum_{\mathcal{T}} \int_{\mathbf{c}_{\mathcal{S}}} \Big(\prod_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}}) q(\mathbf{c}_{\mathcal{S}}) \Big) \Big(\log p(\mathbf{v}_i | \{\mathbf{c}_{\mathcal{S}}\}) + \log p(\mathbf{c}_{\mathcal{S}} | \mathcal{T}_{\mathcal{S}}) \Big) \\
0 &= \frac{\partial}{\partial \nu_i} \sum_{\mathcal{S}, k:\mathcal{S}(k)=i} \sum_{\mathcal{T}_{\mathcal{S}}} q(\mathcal{T}_{\mathcal{S}}) \Big(\log p(\mathbf{v}_i | \{\nu_{\mathcal{S}}\}) + \log e_{\mathcal{T}_{\mathcal{S}}(k)}(\nu_{\mathcal{S}(k)}) \Big) \\
0 &= \frac{\partial}{\partial \nu_i} \Big((\mathbf{v}_i - \nu_i)^2 / 2\sigma_i^2 + \sum_{\mathcal{S}, k:\mathcal{S}(k)=i} \sum_{\mathcal{T}_{\mathcal{S}}} q(\mathcal{T}_{\mathcal{S}}) (\nu_i - \mu_{\mathcal{T}_{\mathcal{S}}(k)})^2 / 2\phi_{\mathcal{T}_{\mathcal{S}}(k)} \Big) \\
0 &= -(\mathbf{v}_i - \nu_i)/\sigma_i^2 + \sum_{\mathcal{S}, k:\mathcal{S}(k)=i} \sum_{\mathcal{T}_{\mathcal{S}}} q(\mathcal{T}_{\mathcal{S}}) (\nu_i - \mu_{\mathcal{T}_{\mathcal{S}}(k)}) / \phi_{\mathcal{T}_{\mathcal{S}}(k)},
\end{aligned}$$

gathering and isolating variables gives the following result:

$$\nu_{x,y,t} \leftarrow \frac{\frac{\mathbf{v}_{x,y,t}}{\sigma^2_{x,y,t}} + \sum_{\mathcal{S}, k:\mathcal{S}(k)=(x,y,t)} \sum_{\mathcal{T}_{\mathcal{S}}} q(\mathcal{T}_{\mathcal{S}}) \frac{\mu_{\mathcal{T}_{\mathcal{S}}(k)}}{\phi_{\mathcal{T}_{\mathcal{S}}(k)}}}{\frac{1}{\sigma^2_{x,y,t}} + \sum_{\mathcal{S}, k:\mathcal{S}(k)=(x,y,t)} \sum_{\mathcal{T}_{\mathcal{S}}} q(\mathcal{T}_{\mathcal{S}}) \frac{1}{\phi_{\mathcal{T}_{\mathcal{S}}(k)}}}.$$

Optimizing for the epitome parameter μ_j, $j = (x_e, y_e, t_e)$, is done in a similar manner:

$$\begin{aligned}
\frac{\partial F}{\partial \mu_j} &= -\frac{\partial}{\partial \mu_j} \sum_{\mathcal{S}} \sum_{\mathcal{T}} \int_{\mathbf{c}_{\mathcal{S}}} \Big(\prod_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}}) q(\mathbf{c}_{\mathcal{S}}) \Big) \log p(\mathbf{c}_{\mathcal{S}} | \mathcal{T}_{\mathcal{S}}) \\
0 &= \frac{\partial}{\partial \mu_j} \sum_{\mathcal{T}, k:\mathcal{T}(k)=j} \sum_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}} = \mathcal{T}) \log p(\nu_{\mathcal{S}(k)} | \mathcal{T}_{\mathcal{S}} = \mathcal{T}) \\
0 &= \frac{\partial}{\partial \mu_j} \sum_{\mathcal{T}, k:\mathcal{T}(k)=j} \sum_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}} = \mathcal{T}) (\nu_{\mathcal{S}(k)} - \mu_j)^2 / 2\phi_j \\
0 &= \sum_{\mathcal{T}, k:\mathcal{T}(k)=j} \sum_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}} = \mathcal{T}) (\nu_{\mathcal{S}(k)} - \mu_j) / \phi_j
\end{aligned}$$

$$\mu_{x_e,y_e,t_e} \leftarrow \frac{\sum_{\mathcal{T}, k:\mathcal{T}(k)=(x_e,y_e,t_e)} \sum_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}} = \mathcal{T}) \nu_{\mathcal{S}(k)}}{\sum_{\mathcal{T}, k:\mathcal{T}(k)=(x_e,y_e,t_e)} \sum_{\mathcal{S}} q(\mathcal{T}_{\mathcal{S}} = \mathcal{T})}.$$

The steps in deriving the update equation for the epitome parameter ϕ_j, where $j =$

(x_e, y_e, t_e), follows the same initial steps as above:

$$\frac{\partial F}{\partial \phi_j} = \frac{\partial}{\partial \phi_j} \sum_{\mathcal{T}, k:\mathcal{T}(k)=j} \sum_{\mathcal{S}} q(\mathcal{T}_\mathcal{S} = \mathcal{T}) \Big((\nu_{\mathcal{S}(k)} - \mu_j)^2 / 2\phi_j + \frac{1}{2} \log \phi_j \Big)$$

$$0 = \sum_{\mathcal{T}, k:\mathcal{T}(k)=j} \sum_{\mathcal{S}} q(\mathcal{T}_\mathcal{S} = \mathcal{T}) \Big(-(\nu_{\mathcal{S}(k)} - \mu_j)^2 / 2\phi_j^2 + \frac{1}{2\phi_j} \Big)$$

$$\phi_{x_e, y_e, t_e} \leftarrow \frac{\sum_{\mathcal{T}, k:\mathcal{T}(k)=(x_e,y_e,t_e)} \sum_{\mathcal{S}} q(\mathcal{T}_\mathcal{S} = \mathcal{T}) (\nu_{\mathcal{S}(k)} - \mu_{x_e,y_e,t_e})^2}{\sum_{\mathcal{T}, k:\mathcal{T}(k)=(x_e,y_e,t_e)} \sum_{\mathcal{S}} q(\mathcal{T}_\mathcal{S} = \mathcal{T})}.$$

5.5 Trading off space and time

The video epitome can compress a video both spatially and temporally. The size of the epitome acts as a knob that can be tuned to adjust the amount of compression in both space and time. The balance between space and time has a profound effect on the resulting video epitome and how it models the video.

Fig. 5.3a shows a few frames from a sample video where a toy car drives around a mat. On one extreme, the video epitome can emphasize spatial compression, as in Fig. 5.3b. In this circumstance, individual motion patterns are isolated and several frames in the epitome are dedicated to each motion. Note that the epitome is taken over a torus, that is, the epitome circularly wraps at the edges in order to maximize the use of all the available pixels. Conversely, Fig. 5.3c shows a video epitome that greatly compresses the time dimension of the video. With just a few frames to work with, the video epitome models multiple motion patterns simultaneously within its frames.

Usually the mean of the epitome is initialized by sampling each pixel in the epitome independently from a gaussian distribution with the same mean and variance as that of the original video sequence. An inherent structure can be imposed upon the epitome by using a more structured initialization. For example, the epitome in Fig. 5.3d has

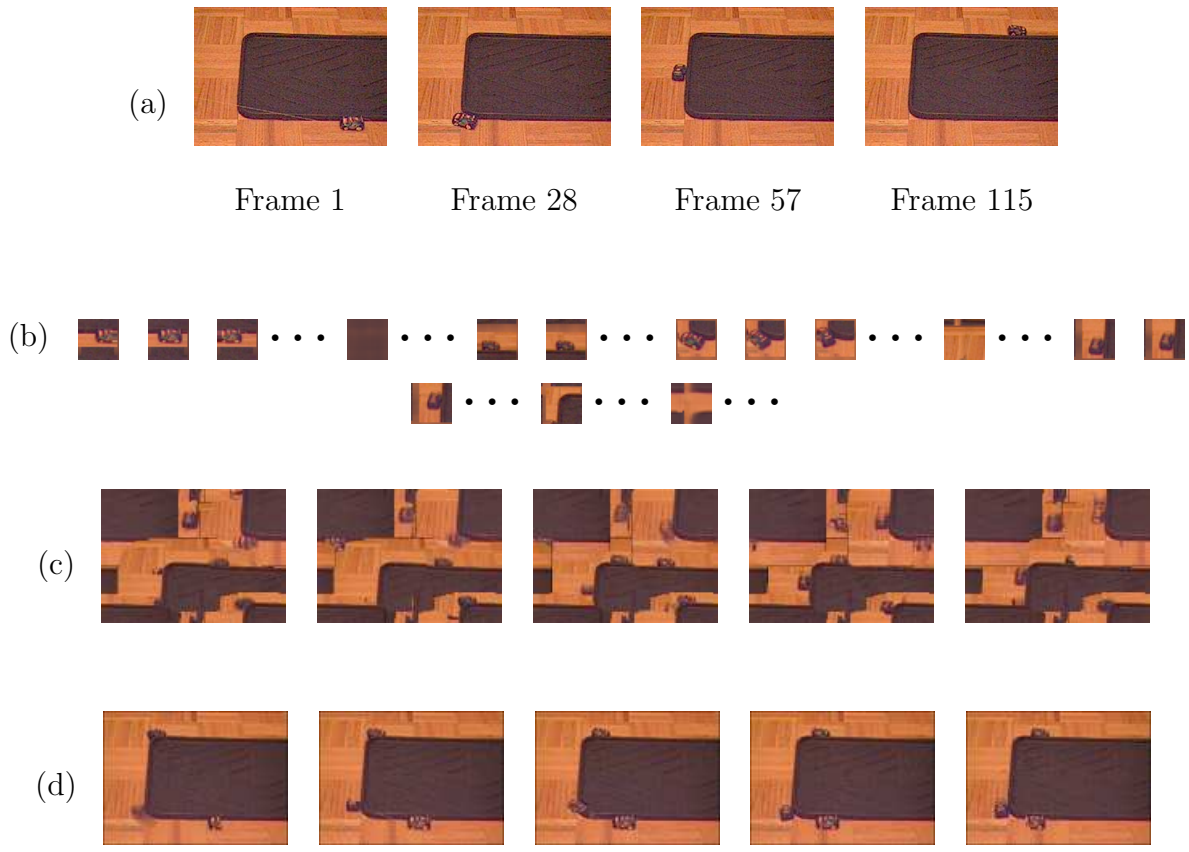

Figure 5.3: Varying the dimensions of the epitome. (a) A few frames from a 208x288x115 video sequence of a toy car driving around a mat. (b) Several frames from its 60x60x80 video epitome, demonstrating an epitome that is spatially small. (c) A 208x288x5 video epitome, which is on the other extreme and significantly compresses time. (d) A 208x288x5 video epitome initialized in a more structured manner to encourage spatial features to align to those in the original sequence. All these epitomes contain approximately the same number of pixels and are each more than 20 times smaller than the original video. A variety of patch sizes were used in learning the epitome, varying from two to four along the time dimension and 10x20 to 15x25 and 15x10 to 20x12 spatially.

the same dimensions and was learnt in the same manner as the epitome in Fig. 5.3c, but the spatial structures are aligned with those in the original sequence. This effect was accomplished by initializing each pixel in the epitome to be the median value of the pixels across all the frames in that exact spatial location, plus a small amount of noise. The posterior epitome responsibility, $q(\mathcal{T}_{\mathcal{S}})$, from (5.8), then generally favors coordinates such that the spatial locations match the original video sequence because of the high agreement in pixel values.

These video epitomes contain approximately the same total number of pixels, but have much different appearances. However, in all of these epitomes, the essential structural and motion components are maintained. While the video epitome can itself be useful for visual purposes, its true power arises when used within a larger model for performing various video processing applications as described in the next section. When used for a data processing application, the dimensions of the epitome should achieve a balance between space and time so that both large spatial and temporal features can be modeled.

5.6 Epitomic video processing

First, we note that a very high variance parameter $\sigma^2_{x,y,t}$ effectively severs all the video cubes $\mathbf{c}_{\mathcal{S}}$ from the observation $\mathbf{v}_{x,y,t}$. According to the generative model equation (5.3), an excessive level of noise $\sigma^2_{x,y,t}$ will make the generated value $\mathbf{v}_{x,y,t}$ dominated by Gaussian noise and thus independent from the epitome generated video cubes. Because of this, the inferred color $\nu_{x,y,t}$ shared for the coordinate (x, y, t) in all video cubes is dominated by the epitome prediction, as seen in (5.7). When $\sigma^2_{x,y,t}$ is very low, on the other hand, (5.7) is dominated by the observation $\mathbf{v}_{x,y,t}$. This general observation leads to several video reconstruction applications of the inference algorithm described in the previous section, including filling-in missing data, obstruction removal, video interpolation, denoising, and super-resolution.

In a large area of high variance σ^2, iterating (5.7) and (5.8) will fill the inferred values ν with the epitome generated cubes that tend to agree in the overlapping regions. To better understand this, consider randomly initialized set of ν values. The inference step (5.8) for the cube $\mathbf{c}_{\mathcal{S}}$ takes the current guess at $\nu_{\mathcal{S}}$ and evaluates how likely each epitome cube $e_{\mathcal{T}}$ is to generate $\nu_{\mathcal{S}}$. Since the posterior epitome responsibilities share the ν values, the (probabilistically) chosen epitome cubes tend to have a higher level of agreement in the overlaps than if a set of random epitome cubes were selected as responsible for the video cubes. Now, applying (5.7) will replace the initialized values ν with the average votes from the epitome cubes deemed likely to have been generated according to $q(\mathcal{T}_{\mathcal{S}})$. Since the likely video cubes have a moderate level of agreement, the generated ν texture will now be more consistent with the epitome texture. Iterating these two steps will lead to the solution where $\mathcal{T}_{\mathcal{S}}$ are chosen so as to "quilt" a random, but consistent texture from the epitome. Note also, that in these steps, each new estimate for a $\nu_{x,y,t}$ is a weighted average of *all* epitome means μ_{x_e,y_e,t_e} weighted by both the inverse epitome variances ϕ_{x_e,y_e,t_e} and the probabilities of all possible cube mappings that lead to mapping epitome entry (x_e, y_e, t_e) to video entry (x, y, t). After several iterations, however, the $q(\mathcal{T}_{\mathcal{S}})$ become fairly peaked and consistent.[1]

If the area of high variance σ^2 is surrounded by the area of low variance, then the quilting steps described above will be automatically conditioned on satisfying the agreement of the ν values in the areas of low variance. Thus, we can perform video repair operations by setting σ^2 values to be high in the areas of video that are either missing, or considered bad, or that we simply want to replace with the texture from the epitome.

An interesting additional property of the algorithm in the previous section is that the

[1]For example, if the responsible epitome cube for the video cube at $\{1, ..., 5\} \times \{1, ..., 5\} \times \{1, ..., 5\}$ is at epitome coordinates $\{11, ..., 15\} \times \{11, ..., 15\} \times \{11, ..., 15\}$, then the video cube at $\{2, ..., 6\} \times \{3, ..., 7\} \times \{4, ..., 8\}$ will tend to be mapped to epitome coordinates $\{12, ..., 16\} \times \{13, ..., 17\} \times \{14, ..., 18\}$, so that the for instance, the video coordinate $(3, 4, 5)$ maps to the epitome coordinate $(13, 14, 15)$ in both mappings. When the inference procedure does not result in such translational consistency, then it usually results in appearance consistency, i.e., each pixel tends to map to similar distributions $e_{x,y,t}$ in all mappings.

epitome learning can be performed *jointly* with the repair operations, even when some of the input video pixels are known to be unreliable. If the unreliable (or unwanted) pixels are given high noise parameters σ^2, then iterating all four of the equations (5.7-5.10) will construct the epitome only from the reliable pixels. (Only the reliable pixels propagate into ν values, which in turn define the epitome statistics in each step (5.10)).

Next, we define several video reconstruction tasks by using a given $X \times Y \times T$ video $\mathbf{f}_{x,y,t}$ to form the input $\mathbf{v}_{x,y,t}, \sigma^2_{x,y,t}$ to the epitome inference algorithm, with all parameters initialized in an uninformative way. We consider the resulting set of parameters ν a video reconstruction. In comparison to "texture quilting" techniques used in [14] for example, epitome offers both better generalization and potentially greater computational efficiency. The cost of learning an epitome is proportional to the product of the data size and the epitome size, while the cost of the quilting operations is proportional to the product of the library size and the reconstruction size. If the epitome is much smaller than the training data, then the total cost of using the epitome as the library becomes lower than the cost of using the entire training video as the library. Furthermore, epitomes can be used as a set of pointers to the original training data, so that upon the computation of the posterior $q(\mathcal{T}_\mathcal{S})$, the training data cubes corresponding to the several best matching epitome cubes $\mathcal{T}_\mathcal{S}$ can be searched over to find potentially even better candidates for quilting. This offers the advantage of using all the original training cubes in quilting while still using the computational benefit and regularization of the epitome representation.

5.6.1 Denoising

If the noise level σ^2 in the given video $\mathbf{f}$ is known and constant, then we set $\mathbf{v} = \mathbf{f}$ and $\sigma^2_{x,y,t} = \sigma^2$ and iterate (5.7-5.10), starting with non-informative initialization. This procedure effectively performs denoising by averaging similar pieces of the video. The video epitome serves as a nexus of self-similarity estimation, and iteration is necessary to estimate the epitome jointly with the denoising operation. The size of the epitome is

critical in this operation. If the epitome is assumed to be very large, then, the denoising operation may still copy a lot of noisy pixels into the epitome. On the other hand, a small epitome will be forced to be more liberal in the definition of similarity, due to the lack of resources to capture the diversity. Averaging many video cubes into a small epitome may lead to excessive blurring.

If the noise level σ^2 is not known, but is assumed to be *uniform* in the video, it can be estimated from the data by adding the following step to the algorithm

$$\sigma^2 = \frac{1}{XYT}\sum_{x,y,t}(\nu_{x,y,t} - \mathbf{v}_{x,y,t})^2, \tag{5.11}$$

which follows from setting to zero the derivative of F with respect to σ^2.

5.6.2 Super-resolution

Suppose that we want to increase the video resolution of the given video $\mathbf{f}_{x,y,t}$. To guide us in this task, let us assume that we have a high resolution video $\mathbf{h}_{x,y,t}$ of a similar scene or scenes. We can then iterate (5.7-5.10) using $\mathbf{v} = \mathbf{h}$ and small noise levels $\sigma^2_{x,y,t} = \epsilon$. If $\mathbf{h}$ is assumed to be somewhat noisy, we can increase $\sigma^2_{x,y,t}$ or learn it as described in the previous subsection. The resulting high resolution epitome e can now be used to iterate only (5.7) and (5.8) on the input video defined as:

$$\begin{aligned}\mathbf{v}_{x,y,t} &= \mathbf{f}_{x/n,y/n,t}\\ \sigma^2_{x,y,t} &= \epsilon + \sigma^2[mod(x,n) > 0][mod(y,n) > 0],\end{aligned} \tag{5.12}$$

where ϵ is a small number and σ^2 is large, and [] is an indicator function. In other words, $\mathbf{v}$ is a $nX \times nY \times T$ video created by nearest-neighbor resolution enhancement of $\mathbf{f}$, but the map $\sigma^2_{x,y,t}$ labels as unreliable (noisy) the pixels which do not have coordinate divisible by n. Iterating (5.7) and (5.8) will replace the unreliable pixels with the best matching texture of e learned from $\mathbf{h}$.

The reader can appreciate that a similar approach can be utilized to fill in the missing "in-between" pixels even if the missing pixels are not uniformly inserted as above.

5.6.3 Object removal and general missing data reconstruction

We now consider the general case of missing value reconstruction. Define the set of coordinates $\mathcal{G}$ for which the measurements in $\mathbf{v}$ are considered good, and the set of coordinates $\mathcal{M}$ for which the measurements are missing, unreliable, or need to be replaced for any reason. When learning the epitome, the set of training video cubes $\{\mathcal{S}\}$ is chosen so that none of the cubes overlap with missing data. Then, by setting variances $\sigma^2_{\mathcal{G}} = \epsilon$ to be small and $\sigma^2_{\mathcal{M}} = \sigma^2$ to be large, the algorithm given by iterating (5.7), (5.8), (5.9), and (5.10), will produce the reconstruction $\nu_{\mathcal{M}}$, quilted from the epitome which captures the patterns in $\mathbf{v}_{\mathcal{G}}$. For example, $\mathcal{M}$ could mark the spatio-temporal segment which has an unwanted object.

5.6.4 Video interpolation

Video interpolation is a special case of the missing data problem and is also similar to that of super-resolution. Here, however, we describe a different version of the problem, one in which a similar video of higher temporal resolution is *not* given.

Formally, we can set up the problem as follows. For a sequence of frames $\mathbf{f}_{x,y,u}$ we know that some frames are missing, so that the given sequence $u = 1, ..., U$ corresponds to the true frames $t(u)$. For instance, for a video interpolation task, we would have that $t = \{1, 3, 5, 7, ...\}$. On the other hand, when a video is broadcast over the Internet, some clients will experience dropped frames in random shorter or longer bursts. Then, t follow a pattern like $t = \{1, 2, 3, 4, 5, 15, 16, 17, 18, 19, 20, 21, 29, 30, 31, 32, ...\}$. For video interpolation tasks, we can define the input to the epitome algorithm as:

$$\begin{aligned} \mathbf{v}_{x,y,t(u)} &= \mathbf{f}_{x,y,u} \\ \sigma^2_{x,y,t(u)} &= \epsilon, \quad \sigma^2_{x,y,t\neq t(u)} = \sigma^2, \end{aligned} \tag{5.13}$$

with ϵ and σ^2 being a small and a large variance respectively. Then equations (5.7-5.10) are iterated to jointly learn the epitome and fill in the missing frames in the reconstruction

ν. When the pattern in t is uniform, as in the case of video interpolation, there is a danger that the epitome will be updated in a similar pattern, for example by decoupling the use of odd and even frames. It is thus generally useful to constrain the epitome learning so that this does not happen. A useful constraint is that the neighboring distributions $e_{x,y,t}$ and $e_{x,y,t+1}$ are similar. In our experiments we simply used potential functions $\Psi_{x,y,t}$ connecting neighboring frames in the epitome as

$$\Psi_{x,y,t} = \max_{x_n,y_n,t_n \in N(x,y,t)} e^{-(\mu_{x,y,t} - \mu_{x_n,y_n,t_n})^2/\psi}, \tag{5.14}$$

where $N(x,y,t)$ denotes a neighborhood of x,y,t outside the frame t, e.g., $N(x,y,t) = \{x-\delta x, ..., x+\delta x\} \times \{y-\delta y, ..., y+\delta y\} \times \{t-1\} \cup \{x-\delta x, ..., x+\delta x\} \times \{y-\delta y, ..., y+\delta y\} \times \{t+1\}$. These potential functions ensure that the neighboring frames are deformed versions of one another, and define the prior

$$p(e) = \frac{1}{Z} \prod_{x_e,y_e,t_e} \Psi_{x_e,y_e,t_e}. \tag{5.15}$$

By including this prior and re-evaluating the derivative of the resulting free energy[2] with respect to the mean $\mu_{x,y,t}$ we can see that (5.9) changes to

$$\mu_{x_e,y_e,t_e} = \frac{\mu_{\hat{x}_{en},\hat{y}_{en},\hat{t}_{en}}/\psi + \hat{\mu}^{\nu}_{x_e,y_e,t_e}/\phi_{x_e,y_e,t_e}}{1/\psi + 1/\phi_{x_e,y_e,t_e}}, \tag{5.16}$$

where $\hat{\mu}^{\nu}_{x_e,y_e,t_e}$ denotes the estimate from (5.9), and $(\hat{x}_{en}, \hat{y}_{en}, \hat{t}_{en}) = \arg\max_{(x_{en},y_{en},t_{en}) \in N(x_e,y_e,t_e)} \Psi_{x,y,t}$.

To perform video interpolation we iterate (5.7),(5.8), (5.16), and (5.10).

Temporal interpolation and super-resolution can be combined in a straight forward manner to derive a super-resolved smooth video from a low-resolution video if it has enough pseudo-repeating structure and the sensor has a short spatio-temporal averaging field.

[2]We ignore the normalization constant Z

5.7 The shifted cumulative sum method

Careful study of the epitome learning rules reveals that many operations are repeated multiple times. For example, a singe pixel $\mathbf{v}_{x,y,t}$ in the video gets evaluated under every epitome distribution e_{x_e,y_e,t_e} several times in different combinations $\mathcal{S},\mathcal{T}$ in which $(x,y,t) \in \mathcal{S}$ and $(x_e,y_e,t_e) \in \mathcal{T}$. Since the pixels are treated as independent, then for each patch of coordinates $\mathcal{S}$, a product of individual distributions is evaluated, or in the log domain, a sum of elements $\log e_{x_e,y_e,t_e}(\nu_{x,y,t})$ is computed. The first observation to make here is that since the epitome algorithm requires that many different overlapping patches are evaluated in this way, it becomes computationally more efficient to compute $\log e_{x_e,y_e,t_e}(\nu_{x,y,t})$ for all combinations of (x_e,y_e,t_e) and (x,y,t) and then sum the appropriate elements for each $\mathcal{S},\mathcal{T}$ combination, than to apply (5.8) directly.

Furthermore, overlapping video cubes even share many of the summations of the $\log e_{x_e,y_e,t_e}(\nu_{x,y,t})$ terms. In fact, if the overlapping cubes $\mathcal{S}$ used in epitome operations are *all* possible cube patches of a given size (or a set of sizes), then these shared summations can be exploited in a systematic way. In particular, consider a *shifted* cumulative sum matrix

$$C(x_s,y_s,t_s) = \sum_{x<x_s,y<y_s,t<t_s} \log e_{\hat{x}_e,\hat{y}_e,\hat{t}_e}(\nu_{x,y,t}),$$

where $\hat{x}_e = (x_e + x_{es})$ *modulus* X_e and similarly for $\hat{y}_e$ and $\hat{t}_e$, which is computed for a particular offset x_{es},y_{es},t_{es}. For the cube patch $\mathcal{T}$ starting at $\mathcal{T}(1) = (x_{es},y_{es},t_{es})$, we can compute at once the epitome likelihoods for all equally shaped cubes $\mathcal{S}$ as a linear combination of up to eight shifted C matrices.[3] This means that for all XYT input cube patches $\mathcal{S}$ of a certain size, and all X_e,Y_e,T_e, the computation of all epitome cube likelihoods is computed in $o(XYTX_eY_eT_e)$ time regardless of the cube size $|\mathcal{S}|$, thus allowing us to work with multiple size video cubes in our experiments, from very

[3]For example, for $\mathcal{T} = \{5,...,10\} \times \{10,...,20\} \times \{2\}$, if we set $(x_{es},y_{es},t_{es}) = (5,10,2)$, the resulting shifted cumulative matrix C leads to $\log e_{\mathcal{T}}(\nu_{[15,20]\times[50,60]\times[17]}) = C(20,60,17) - C(15,60,17) - C(20,50,17) + C(15,50,17)$.

small cubes capturing fine details to very large ones that help strengthen the long-range correlations in the epitome without increasing the computational complexity.

5.8 Experiments

The focus of this section is not of any one specific application, but rather the demonstration of the broad applicability of the model introduced, even without any domain knowledge. The results here show what the epitome model fantasizes of the missing data based on the data used to learn the epitome.

5.8.1 Video Super Resolution

An optical zoom in modern cameras allows the user to trade the field of view for the level of captured detail. The user often desires to capture both the large context of the scene and the detail in it, by first capturing a wide angle shot followed by a large zoom (as shown in the set-up video on http://www.psi.toronto.edu/∼vincent/videoepitome.html) and slow scene scanning at the constant zoom level (omitted in the set-up video for brevity). We can use the approach described in Section 5.6.2 to super resolve the original low resolution wide-angle shot $\mathbf{f}$, using the epitome learnt on $\mathbf{h}$, the high-resolution "scene scanning" video captured at the higher zoom level.

In our example, the wide shot captures a large plant moving in the wind. Later, the camera zooms in and scans the plant creating the shot used for training the high resolution epitome. As described in Section 5.6.2, the high resolution textural and shape features of the plant, as well as the motion patterns, are then used to compute a super-resolved version of the wide shot. Note that the content of the wide shot is similar but far from identical to the content of the high resolution training data. Fig. 5.4 shows a single frame of the super-resolution result compared to bicubic interpolation, and the web page also shows a larger super-resolution sequence of the plant.



low-resolution bicubic interpolation epitome super-resolution

Figure 5.4: Video super resolution. (Left) A single frame from a low resolution 135x90x32 sequence. (Middle) The 360x240 bicubic interpolation of the frame. (Right) The 360x240 video epitome super-resolution result. A 325x225x8 video epitome was learnt from a 360x240x17 high resolution sequence captured when the camera physically zoomed into a similar scene at a different point in time. Super-resolution was performed by reconstructing the low resolution sequence with the epitome containing the features from the high resolution sequence. Video cubes of size 75x75x4, 50x50x3, and 25x25x2 were used during learning and reconstruction. The video of this super-resolution result is available at http://www.psi.toronto.edu/∼vincent/videoepitome.html.

5.8.2 Reconstructing dropped frames from a video broadcast

Real-time streaming videos are becoming increasingly popular on the web. In streaming video, often video frames are dropped because of a lack of bandwidth. Client-side recovery of these missing frames using only the successfully received frames would produce a smother playback to give the appearance of undropped frames.

Fig. 5.5 shows a few frames from a video sequence where the dropped frames effect was simulated and the missing frames were then reconstructed using the video epitome as described in Section 5.6.4. The video epitome is able to consolidate the various disconnected frames and assemble a comprehensible set of motion patterns in the video sequence. In the video on the web page, we show the received frames on the left (with freezes during the frame drops) and on the right, the video reconstruction using only the video on the left as the epitome input. Note that due to the large number of missing

frames, most original motion patterns have never been seen by the epitome algorithm in their uncorrupted form anywhere in the received broadcast. Exemplar-based approaches, while able to fill in some missing frames provided that similar motions appear elsewhere in the video sequence, cannot handle this situation, as nearly every video cube in the sequence contains missing data.

5.8.3 Video in-painting

Video epitomes contain the video's basic structural and motion characteristics, which are useful for in-painting applications. The goal of video in-painting is to fill in missing portions of a video, which can arise with damaged films or occluding objects. The pixels are filled by fantasizing the missing pixels in a manner that is consistent with the rest of the video. Video in-painting can be performed with the video epitome as described in Section 5.6.3.

Fig. 5.6 and its corresponding video on the web show the results of in-painting on a video of a girl walking. The results here are similar to those of [40]. Part way through the video, the girl is occluded by a fire hydrant. Removing the fire hydrant can be formulated as a video reconstruction problem by considering these pixels as missing, learning the video epitome only on the observed pixel values, and performing epitome inference while setting the variances, σ^2 to be high in the area occupied by the hydrant. The video epitome is able to compress the basic walking motion into several frames and transfers this motion pattern into the missing pixels.

5.8.4 Denoising

In Section 5.6.3, we discussed the general case of filling in arbitrary missing data in video given only the corrupted video. Fig. 5.7 illustrates the potential power of the video epitome. A highly corrupted video was created in which each of the RGB color channels of each pixel was missing with 50% probability. The known bad channels for

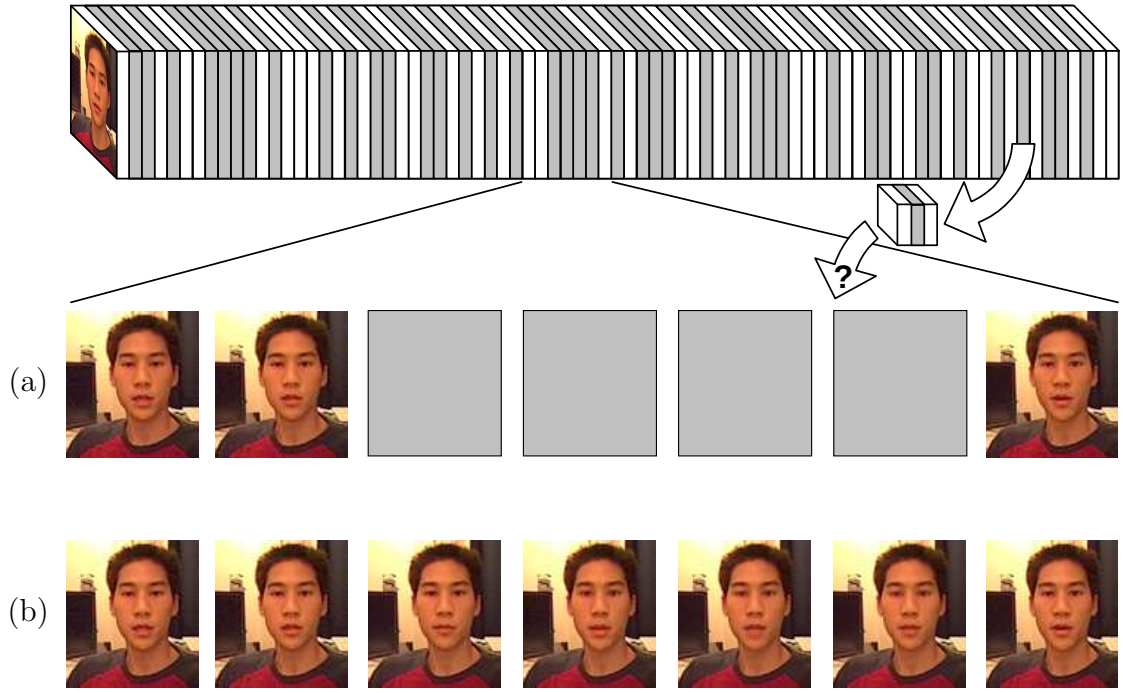

Figure 5.5: Dropped-frames experiment. This experiment deals with the problem of reconstructing missing frames when receiving streaming video over the Internet. The problem is illustrated by the diagram at the top, where gray indicates frames that were dropped during transmission. (a) Several frames from the sequence in which four of the frames are missing. Most video cubes from the sequence contain missing frames, so they cannot simply be stitched together to fill in the missing frames as with exemplar-based methods. (b) The reconstructed missing frames using the video epitome of the sequence with dropped frames. The arrival time of the frames in this 90x100x39 sequence was modeled as a discrete time Bernoulli process whereby the inter-arrival time of the frames was governed by a geometric distribution. Frames were expected to arrive in each time slot and any unfilled time slots between arrived frames were considered to be dropped frames. The mean number of missing frames between arrived frames was one. A 90x100x13 video epitome was learnt from only the non-dropped frames using video patches of size 63x70x6, and 36x40x5, and 30x30x4. This video sequence is available at http://www.psi.toronto.edu/∼vincent/videoepitome.html.

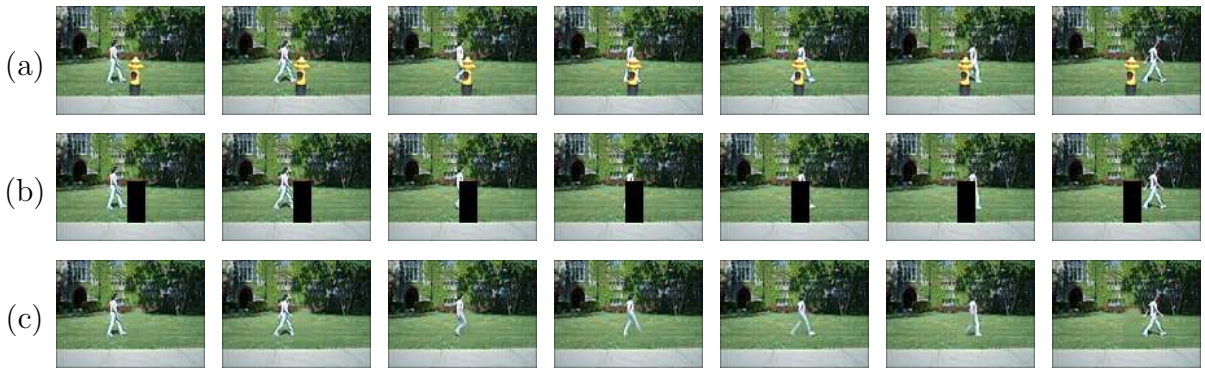

Figure 5.6: Video in-painting experiment using the video epitome. (a) Several frames from a 115x83x53 video. (b) Removal of the fire hydrant from the video by considering these pixels as missing. (c) In-painted frames using a 30x30x15 epitome with 20x20x5, 15x15x4, 10x10x3, and 5x5x2 video cubes. Video available at http://www.psi.toronto.edu/∼vincent/videoepitome.html.

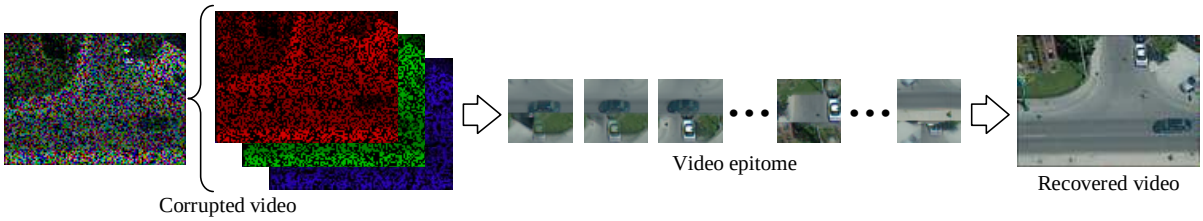

Figure 5.7: Missing channels experiment. A 115x83x24 corrupted video sequence where each of the RGB color channels of each pixel was missing with 50% probability was recovered by learning its video epitome and filling in the missing values using the epitome. The 40x40x12 epitome was learnt from only the observed values and does not contain missing data because of the consolidation of the 20x20x4, 15x15x3, and 10x10x2 video cubes from the video into the smaller video epitome during learning. Video available at http://www.psi.toronto.edu/∼vincent/videoepitome.html.

each pixel were marked by high noise variances σ^2. The video was then given to the epitome learning algorithm and then reconstructed by iterating (5.7-5.10). Despite high levels of corruption, the repetitive motion in the video helped the epitome learn a clean representation and reconstruct the video ν. This video is available on the web page.

5.9 Conclusion

An epitome represents many of the high-order statistics in array data using a more compact array, and is amenable to faster searching and better generalization, compared to patch libraries. Here, we developed an efficient algorithm for learning video epitomes and performing various inference tasks. We illustrated some of the applications of video epitomes, including video enhancement and editing tasks. Videos demonstrating a variety of applications are available at http://www.psi.toronto.edu/∼vincent/videoepitome.html.

In comparison to reconstruction techniques based on a library of known data patches, an advantage of epitomes is that they can be trained directly on corrupted or degraded data, as long as the data is repetitive. Video tends to be highly redundant and is thus quite well-suited to analysis using epitomes. Despite the fact that the video epitome model described in this chapter was only made invariant to translations, it is able to accommodate minor variances in rotation, scale, illumination, colour, and occlusions. Specifically adapting the mapping function to account for these other transformations would allow the model to take advantage of even more redundancies in videos. Because epitomes provide a representation that retains the natural flow of the input data, and offer significant computational and statistical advantages over patch libraries, we believe they will find many uses in data analysis and computer vision.

Chapter 6

Long Range Correlations

6.1 Introduction

Patches have been used to capture local correlations between pixels in various low-level vision tasks, with perhaps the most notable early example in [16]. To capture correlations that span a longer range, larger patches can be used, though this has adverse side-effects, including for example, increased difficulty in patch matching. When multiple patches from different images are matched there are many correlations between the mapping pairs. If two patches match in two images, then it is likely that shifting both patches one pixel in the same direction also leads to a matched pair, since there is a significant amount of overlap between the pixels in the patches. Such local coherence ideas have been used in [3, 40]. In the case of videos and 3D patches, local correlations are even stronger because of the added time dimension.

Visual data also exhibits strong long range correlations in images which can relate patches that do not have any pixels in common. Elastic matching, eg. [7], has been used in the past to register image pairs by leveraging the fact that mappings are generally smooth between images to overcome erroneous correspondences due to noise or lack of identifying features. By reducing the analysis to a subset of patches with relatively high

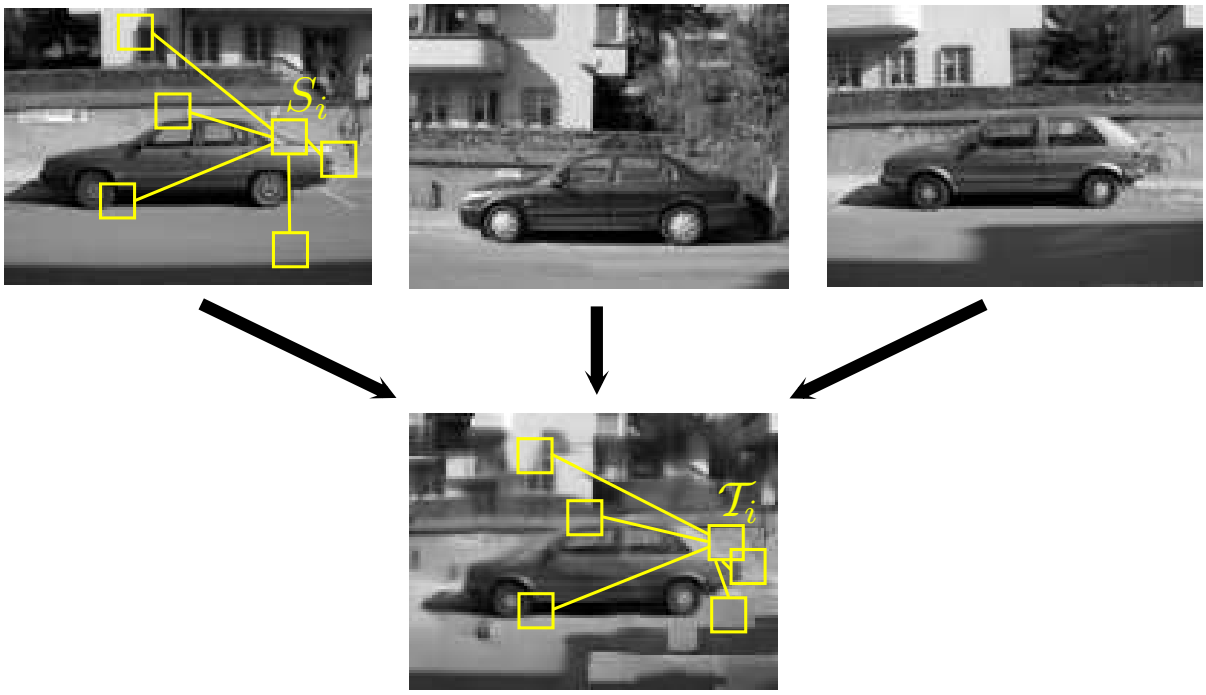

Figure 6.1: The epitome of three car images learnt using long-range patch correlations. The epitome ends up being a merge of the three cars, with both the front and back of the car reaching a compromise between the different shapes of the cars. Also interesting to note is how the epitome merges the three backgrounds. The patch indicated by $\mathcal{S}_i$ in the top-left image is connected to several other randomly chosen patches in the image, to which, relative patch distances should be generally maintained during patch matching to the epitome. The corresponding patch in the epitome is shown as $\mathcal{T}_i$ and the matching is constrained by the matches for the patches connected to $\mathcal{S}_i$. A video illustrating this patch correspondence during learning is available at http://www.psi.toronto.edu/~vincent/patchcorr.html.

mutual distances, it is possible to produce elastic matching of large structures using a small number of image features. This is achieved by assuming that relative offsets of image features are only slightly perturbed between two images. Relative positions of features are also useful for object recognition. For example, in constellation models [9, 39], the relative locations of a small number of detected features from an image are

used to facilitate object recognition. Elastic image matching, which only some of its many forms have been mentioned here, has been one of the most used tools in vision.

In this chapter, we are concerned with the use of similar elastic constraints, but with the goal of modeling correlations among the mappings of all data patches to a common learned representation of a category of images (Fig. 6.1), eg. an epitome [10, 25, 27]. Thus, the model we propose captures the full probability distribution of the data, making it possible to mine the long-range image correlations in various inference tasks, including data registration, data likelihood computation (for tasks such as classification or detection), and missing data interpolation.

The focus of this chapter is on this model and we demonstrate its broad applicability on several tasks. For example, one of the tasks we can perform using inference in our model is the simulation of illumination changes on an object in a single photograph. The illumination training data consists of video sequences of other static objects and varying illumination angles, and the patches mapped to a common epitome are three-dimensional. Our model estimates the appearance and mapping constraints through space and time among the video cubes in the training data and then estimates a video sequence which satisfies these constraints and whose central frame is equal to the given photograph. This creates plausible illumination changes on the object in the photograph.

Previously, such image relighting tasks typically required an expensive, brute force, hardware solution as in [12], where the subject sits in a dome and photos of the subject are taken from many different angles, from which any illumination can be reconstructed by taking combinations of these images. There are several limitations to this approach including the a priori knowledge of the desired illumination change, so deceased individuals cannot be re-lit; the subject must remain still and be tolerant to strobe lights; and the subject must fit in the dome, so entire scenes cannot be re-lit. Less hardware-dependent and more computation-oriented alternatives to re-lighting an image or video sequence have also proposed. For instance, generic face surface geometry and reflectance

models have been used for re-lighting faces [36, 41]. However, once the problem changes to re-lighting something other than a human face, such as an animal, a piece of cloth, or an entire scene, these approaches become more difficult to follow, as they need object-specific surface geometry models. With the exception of a small number of object categories (perhaps only human faces), such models are fairly rare. The richness of the 3-D face modeling literature is the best indication of the difficulty of acquiring such models. The examples of relighting we show in this chapter are example-based - given a small number of examples (sometimes even just one) of how the image of an object changes with smooth variation of illumination angles, we can construct plausible similar changes on another similar object. This removes the need for full modeling of the surface geometry of the objects. Instead, the correlations in the patches form the training data provide sufficient constraints to infer plausible image changes due to illumination angle changes.

In addition to face and cloth photograph-relighting, we show results on simulating a walk through a hallway given one photograph of a hallway, and learning epitomes of cars and faces, all using the same trainable model of data patches.

6.2 Flexible patch configurations

As discussed in the introduction, the issue of varying geometric configurations of object features has repeatedly been encountered in vision research. In this chapter we are particularly concerned with how this variability can be accounted for in patch models that describe learnable probability density functions of images. In particular, as described in Fig. 6.1, we construct an epitome model in which patches from different locations in the image have correlated mappings to the epitome locations. While the discussion in this section is limited to 2-D images for concreteness, it is trivial to extend these ideas to N-D structures.

The original epitome model [25] proposes that a set of pixels from image z with indices

in the set $\mathcal{S}$, i.e., the set $z_{\mathcal{S}} = \{z_{\mathbf{u}} | \mathbf{u} \in \mathcal{S}\}$[1], can be described by specific individual probability distributions taken from epitome (e) locations in the set $\mathcal{T}$:

$$p(z_{\mathcal{S}}|e_{\mathcal{T}}) = \prod_k p(z_{\mathcal{S}(k)}|e_{\mathcal{T}(k)}), \tag{6.1}$$

or simply,

$$p(z_{\mathcal{S}}|e_{\mathcal{T}}) = \prod_k e_{\mathcal{T}(k)}(z_{\mathcal{S}(k)}), \tag{6.2}$$

where it is assumed that the sets $\mathcal{S}$ and $\mathcal{T}$ are ordered and of equal sizes[2], and the k-th index in one set corresponds to the k-th index in the other. Given a number of these correspondences between different subsets of pixels in training images $\mathcal{S}_i$ and subsets of epitome locations $\mathcal{T}_i$, learning an optimal epitome reduces to assembling the required sufficient statistics.

The rules of establishing pixel correspondence (choosing various image locations $\mathcal{S}$ and their corresponding epitome locations $\mathcal{T}$) are left general in these early papers, although particular applications usually considered regular small patches of image pixels to form various sets $\mathcal{S}_i$, and the same size patches in the epitome. This made the search for optimal mapping of each image patch linear in the size of the epitome, as effectively, only the position of the epitome patch is required to fully describe the mapping regardless of the patch size. This choice also limited the spatial extent in which image correlations are nicely captured by the epitome to several patch sizes. Due to the overlap of patches both in the input image(s) and in the epitome, the textures that form in the epitome upon learning capture structures larger than the patch sizes, but often much smaller than the object size.

The basic formulation of the model allows the pixel coordinates in $\mathcal{S}_i$ to come from disconnected parts of the image, and the mapping rules that limit the space of possible

[1]Boldcase $\mathbf{u}$ and $\mathbf{v}$ represent 2D indices describing image coordinates, i.e., $\mathbf{u} = (x, y)$

[2]For example, one way to limit the space of allowed correspondence is to consider subsets $\mathcal{S}_i$ in the data that are rectangular patches of a certain size, i.e., $\mathcal{S}_i = \{\mathbf{u} = (x, y) | X_i \leq x < X_i + \delta, Y_i \leq y < Y_i + \delta\}$ and the corresponding epitome subsets $\mathcal{T}$ are defined to also be rectangular patches starting at some epitome location X_j, Y_j.

sets of epitome coordinates $\mathcal{T}$ to include rotation, shearing, and other transformations. This would allow capturing more complex geometric changes that span a larger spatial extent in the image. To the best of our knowledge, while the inclusion of more sophisticated geometric transformations has been studied before, the use of non-contiguous patches has not been investigated due to the explosion of the numbers of possible image subsets $\mathcal{S}_i$ to be considered. Recently, patches of arbitrary (and inferred) shape have been used in epitome structures dubbed jigsaws [27], but these patches are still contiguous and do not capture global correlations in images. Without directly capturing longer range correlations in the data, be it images, videos, or other ordered datasets, the epitome models will fail to capture global scale phenomena of the objects they were trained on.

To resolve this problem, instead of using non-contiguous patches to capture within each single mapping, $\mathcal{S} \rightarrow \mathcal{T}$, the correlations in distant parts of the image, we propose to model correlations among *different* mappings, $\mathcal{S}_i \rightarrow \mathcal{T}_i$. This allows us to capture long-range correlations in the image while still having relatively simple individual patches and mappings.

6.2.1 The mapping field

The use of simple rectangular patches to represent data has significant computational advantages, especially for higher dimensional data, as discussed, for example, in [10]. Rectangular patches allow the use of fast Fourier transform tricks and efficient image correlation computations necessary to efficiently perform otherwise very expensive computations. Smaller patches of other shapes can be simulated using the masking variables [25], or, with a higher computational cost, but some other benefits, using jigsaw models [27]. Different patches of data coordinates $\mathcal{S}_i$ have the associated mapped epitome coordinates $\mathcal{T}_i$. The original epitome model assumed independence of variables $\mathcal{T}_i$, as the patch overlap naturally enforced the appropriate agreements in mappings of nearby patches. Similar local agreement is enforced in the jigsaw model in a way that allows

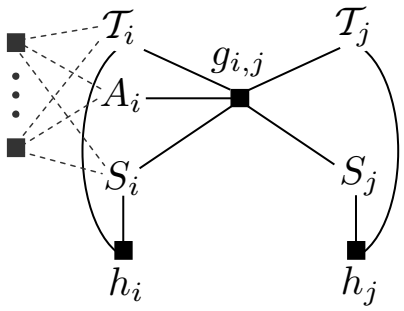

Figure 6.2: Factor graph of the long-range correlations patch model.

patches to be arbitrarily shaped. In the model we propose in this chapter, we capture the constraints on the mappings $\mathcal{T}_i$ and $\mathcal{T}_j$ of *distant* patches $\mathcal{S}_i$ and $\mathcal{S}_j$ through agreement factors $g_{i,j} = g(\mathcal{T}_i, \mathcal{T}_j, \mathcal{S}_i, \mathcal{S}_j)$ (Fig. 6.2), which have high value if the mappings $\mathcal{T}_i$, $\mathcal{T}_j$ keep a similar geometric configuration as $\mathcal{S}_i$, $\mathcal{S}_j$. The factors h_i correspond to the usual epitome likelihoods eg. $h_i = e_{\mathcal{T}_i}(\mathcal{S}_i)$. Intuitively, this is represented in Fig. 6.1, where the patches connected to $\mathcal{S}_i$ constrain its matching to $\mathcal{T}_i$ as it is desirable not only for the two patches to be similar, but also to maintain the relative locations of the matching patches. The likelihood of the entire image is proportional to the product of all factors (only some of which are shown in Fig. 6.2),

$$p(z_{\mathcal{S}_1}, z_{\mathcal{S}_2}, ..., z_{\mathcal{S}_I}) \propto \prod_{i=1}^{I} h_i \prod_{j \in N_i} g_{i,j}, \tag{6.3}$$

where I is the total number of image patches considered, and N_i represents the set of data patches j connected to patch i in the model. While this set can be arbitrary for each patch i, in our experiments we chose a particular (but randomly chosen) relative configuration and use it for all patches in the image.

There is a number of ways to parameterize the relative geometric configuration of the patches, and some alternatives we have not tested will be discussed later, but first, we go over the choice of factors g and long-range interaction neighborhoods N in our experiments. The basic property that factors g are enforcing is that the relative positions of the coordinates in $\mathcal{S}_i$, $\mathcal{S}_j$ are preserved in the mappings $\mathcal{T}_i$, $\mathcal{T}_j$, i.e., $\mathcal{S}_i(k) - \mathcal{S}_j(k) \approx$

$\mathcal{T}_i(k) - \mathcal{T}_j(k)$[3]. If each patch is kept rectangular, this is equivalent to $\bar{\mathcal{S}}_i - \bar{\mathcal{S}}_j \approx \bar{\mathcal{T}}_i - \bar{\mathcal{T}}_j$, where bar denotes taking the mean of the coordinates in the set, since $\Delta\mathcal{S} = \mathcal{S}_i(k) - \mathcal{S}_j(k)$ is constant for all elements, and the same is true for $\Delta\mathcal{T}$. If the mapping inference enforces a preference to keeping the relative positions of the chosen patches, the epitomes would reflect longer-range correlations in images. However, the images often undergo geometric deformations due to angle of view changes and object deformations, which can violate some of these constraints, and to account for that, we can allow for different variances on the Gaussians that enforce them, $g_{i,j} = \mathcal{N}(\bar{\mathcal{T}}_i - \bar{\mathcal{T}}_j; \bar{\mathcal{S}}_i - \bar{\mathcal{S}}_j, \mathbf{\Phi}_{i,j})$. In this way, the mappings $\mathcal{S}_j \to \mathcal{T}_j$ for the neighbors of $\mathcal{S}_i$, i.e., $j \in N_i$ will effect the mapping $\mathcal{S}_i \to \mathcal{T}_i$.

In our experiments, the neighborhood N_i consists of K patch (rather than pixel) indices (usually 10-20). There are roughly as many different rectangular patches as there are pixels in the image, since the patch can be centered at any pixel except those close to image boundaries. Thus patches can be indexed by their central pixels. To choose a neighborhood for each patch $S_{\mathbf{i}}$, where $\mathbf{i}$ now represents a 2-D coordinate of the central pixel, we first choose K *random* 2-D offsets $\mathbf{\Delta}_k$ up to some maximal distance d (eg. half or quarter of the image size), i.e., $||\mathbf{\Delta}_k|| \leq d$ for all k, and define $N_{\mathbf{i}}$ as an ordered set with $N_{\mathbf{i}}(k) = \mathbf{i} + \mathbf{\Delta}_k$. In other words, to construct the field of mapping constraints, each patch i is connected to interacting neighbors in the same relative configuration, but the mapped epitome patches $\mathcal{T}_{\mathbf{j}}$, $\mathbf{j} \in N_{\mathbf{i}}$ may not follow fixed configurations due to the uncertainty captured in the 2-D covariance matrix $\mathbf{\Phi}_{i,j}$ in the Gaussians $g_{i,j}$.

The K Gaussians $g_{i,j}$ for some i should have linked parameters, since they should all depend on the local deformation at i. Furthermore, the assumption $\bar{\mathcal{S}}_i - \bar{\mathcal{S}}_j \approx \bar{\mathcal{T}}_i - \bar{\mathcal{T}}_j$ is too rigid, both because the possible squishing of the texture in the epitome and because of the local image foreshortening and object deformations due to viewing angle changes

[3]While this could be achieved by simply merging the patches $\mathcal{S}_i$ and $\mathcal{S}_j$ into one non-contiguous patch $\mathcal{S}$and imposing constraints on the epitome mapping $\mathcal{T}$, the patches in the epitome may no longer have a fixed shape, thus making it impossible to use cumulative sum and other computational tricks to perform efficient computation of the patch likelihoods h_i for all possible patches $\mathcal{T}_i$.

and other effects. To account for this, we introduce a hidden transformation A_i which affects each of the patch links, i.e., factors $g_{i,j}$,

$$g_{i,j} = \mathcal{N}(\bar{\mathcal{T}}_i - \bar{\mathcal{T}}_j; A_i(\bar{\mathcal{S}}_i - \bar{\mathcal{S}}_j), \mathbf{\Phi}_{i,j}). \tag{6.4}$$

In our experiments this transformation is linear and thus A_i is a matrix. The prior on this matrix can be included so as to prefer identity (not shown in Fig. 6.2). When, as in our experiments, each patch is connected to a large number of interacting neighbors (N_i contains a sufficiently large number of patches), A_i is inferrable. In our experiments we link parameters $\mathbf{\Phi}_{i,j}$ for different patches,

$$\mathbf{\Phi}_{m,N_m(k)} = \mathbf{\Phi}_{n,N_n(k)} = \mathbf{\Phi}_k. \tag{6.5}$$

In other words the links in the same relative configuration (the same $\mathbf{\Delta}_k$) share the same covariance structure. This allows learning the relative extent of the image correlations – the links that tend to lead to low correlation (eg. because they reach too far in some direction) will simply have high variance captured in $\mathbf{\Phi}_k$.

As in some previous patch models, to account for image intensity changes (darkening or brightening of the patches, for example), we add two scalar hidden variables a, b that control the patch contrast in the factors h_i:

$$h_i = e_{\mathcal{T}_i}(a_i z_{\mathcal{S}_i} + b_i). \tag{6.6}$$

6.3 Mapping inference

Next we discuss inference in the epitome model with long-range patch correlations defined by (6.3), (6.4), (6.5) and (6.6). This model is a Markov random field (but unlike in most vision applications with more frequent and further-reaching links), with the epitome as the representation of the observation likelihoods. A number of techniques for inference in MRFs have been studied in the past, and most of them can be adopted

here, including sampling, loopy belief propagation, and variational techniques (for review and some comparisons of probabilistic inference techniques see [18]). We have experimented with a simple variational technique[4], which factorizes the posterior distribution as $Q = \prod_i q(A_i)q(a_i, b_i|\mathcal{T}_i)q(\mathcal{T}_i)$ and further assumes that $q(a_i, b_i|\mathcal{T}_i)$ and $q(A_i)$ are delta functions. The resulting update rules are:

$$\begin{aligned} q(\mathcal{T}_i) &\propto \tilde{h}_i(\mathcal{T}_i) \sum_{\mathcal{T}_j|j\in N_i} \prod_{j\in N_i} q(\mathcal{T}_j) g_{i,j}(\mathcal{T}_i, \mathcal{T}_j, \tilde{A}_i), \\ \tilde{h}_i(\mathcal{T}_i) &= \arg\max_{a,b} h_i(\mathcal{T}_i, a, b), \\ \tilde{A}_i &= \arg\max_{A_i} \sum_{\mathcal{T}_i} q(\mathcal{T}_i) \sum_{\mathcal{T}_j|j\in N_i} \prod_{j\in N_i} q(\mathcal{T}_j) g_{i,j}(\mathcal{T}_i, \mathcal{T}_j, \tilde{A}_i). \end{aligned} \tag{6.7}$$

These equations do not update the belief $q(\mathcal{T}_i)$ about where each patch $\mathcal{S}_i$ should map only according to the epitome likelihoods for different possible patches $\mathcal{T}_i$ as in (2.3). Instead they take into account the probable mappings of the patches in N_i to skew the inference so as to have these patches in the proper geometric configuration with $\mathcal{T}_i$. Using the best matching contrast parameters a, b also allows the inference to be somewhat invariant to illumination changes. Finally, $\tilde{A}_i$ captures shearing of the image as it affects patch $\mathcal{S}_i$. For a diagonal transformation A_i, with diagonal elements A_{i_m}, the update equation is given by

$$\tilde{A}_{i_m} = \frac{\sum_{j\in N_i} \sum_{\mathcal{T}_i} \sum_{\mathcal{T}_j} q(\mathcal{T}_i) q(\mathcal{T}_j) (\bar{\mathcal{T}}_{i_m} - \bar{\mathcal{T}}_{j_m})(\bar{\mathcal{S}}_{i_m} - \bar{\mathcal{S}}_{j_m})}{\sum_{j\in N_i} (\bar{\mathcal{S}}_{i_m} - \bar{\mathcal{S}}_{j_m})^2}. \tag{6.8}$$

Depending on the strength of the links defined by $\mathbf{\Phi}_k$,this shearing may be only local or more global. The update equation to account for uncertainties in the correlation links is given by

$$\mathbf{\Phi}_k = \frac{\sum_{i,j|j\in N_i} \sum_{\mathcal{T}_i} \sum_{\mathcal{T}_j} q(\mathcal{T}_i) q(\mathcal{T}_j) D_{ij}^2}{\sum_{i,j|j\in N_i} 1}, \tag{6.9}$$

[4]Due to a large number of links, belief propagation yields essentially equivalent messages.

where

$$D_{ij} = \bar{\mathcal{T}}_i - \bar{\mathcal{T}}_j - A_i(\bar{\mathcal{S}}_i - \bar{\mathcal{S}}_j).$$

Note that the epitome e involved in computation of h_i can either be learned or preset. For instance, in Fig. 6.5 we simply use an example of a video which we feel sufficiently epitomizes the class of data of interest and define the mean of the epitome e to be equal to that video, and use a small uniform value for all epitome variance. Then, the inference rules above, when iterated can be used to map other videos to it. To also learn the epitome from data, the original update rules, eg. (2.5, 2.6), only need to be changed slightly to account for the contrast variables and reverse the scaling and addition used during matching:

$$\mu_{\mathbf{u}} = \frac{\sum_i \sum_{\mathcal{T}} q(\mathcal{T}_i = \mathcal{T}) \sum_k [\mathbf{u} = \mathcal{T}(k)](z_{\mathcal{S}_i(k)} - b_i)/a_i}{\sum_i \sum_{\mathcal{T}} q(\mathcal{T}_i = \mathcal{T}) \sum_k [\mathbf{u} = \mathcal{T}(k)]}. \tag{6.10}$$

As in the previous work, the epitome update is iterated with the inference equations above.

6.4 Interpolating missing data

In (6.3) we model a selection of data patches. In our experiments, the image patches we considered are all image or video patches of a certain size. In many applications, a model of individual pixels is required, and the fact that each pixel belongs to several patches needs to be resolved. We follow the recipe from [10] and [25] – the patches $z_{\mathcal{S}}$ are in a *hidden* image, while the observed image, at each pixel contains the average of appropriate pixels in all patches $z_{\mathcal{S}}$ that overlap it. The patch agreements are enforced in the inference distribution, rather than in the model. In our case, to the factors h and g described above, we add an extra factor $f_{\mathbf{u}}$ *per pixel* $x_{\mathbf{u}}$ of the observed image x,

$$f_{\mathbf{u}} = \mathcal{N}(x_{\mathbf{u}}; \frac{\sum_i \sum_k [\mathbf{u} = \mathcal{S}_i(k)] z_{\mathcal{S}_i(k)}}{\sum_i \sum_k [\mathbf{u} = \mathcal{S}_i(k)]}, \rho^2_{\mathbf{u}}), \tag{6.11}$$

with the total image likelihood proportional to

$$p(x) \propto \big(\prod_{\mathbf{u}} f_{\mathbf{u}}\big)\big(\prod_{i} h_i \prod_{j} g_{i,j}\big). \tag{6.12}$$

The variational posterior is factorized as $Q = \prod_{\mathbf{u}} q(z_{\mathbf{u}}) \prod_i q(A_i) q(a_i, b_i|\mathcal{T}_i) q(\mathcal{T}_i)$, with a single part of the posterior $q(z_{\mathbf{u}}) = \delta(z_{\mathbf{u}} - \nu_{\mathbf{u}})$ for each particular pixel $z_{\mathbf{u}}$ in the hidden image, regardless of how many patches $z_{\mathcal{S}_i}$ it may be in. This enforces the agreement of overlapping patches in the posterior distribution over all hidden variables. The posterior, as well as model parameters are estimated by minimizing the free energy

$$F = \sum_{hiddens} Q \log \frac{\big(\prod_{\mathbf{u}} f_{\mathbf{u}}\big)\big(\prod_i h_i \prod_j g_{i,j}\big)}{Q}. \tag{6.13}$$

Not only does the model describe the likelihood[5] of image pixels rather than patches (still capturing a number of pixel correlations), but it also makes possible the inference of hidden pixels $z_{\mathbf{u}}$. Inferring these hidden pixels has various applications such as denoising and superresolution as in [10], which are all achieved by setting some of the variances $\rho^2_{\mathbf{u}}$ to large values. However, the inference procedure in our model will involve enforcing long-range correlations in the image. While this property should be helpful in previous applications of patch models, even more ambitious tasks can be attempted – the ones for which accounting for long range correlations in the data is crucial. Some of these tasks will be illustrated in the experimental section.

The inference of the hidden image pixels $z_{\mathbf{u}}$ reduces to estimation of parameters $\nu_{\mathbf{u}}$:

$$\nu_{\mathbf{u}} = \frac{\frac{x_{\mathbf{u}}}{\rho^2_{\mathbf{u}}} + \sum_{i,k|\mathcal{S}_i(k)=\mathbf{u}} q(\mathcal{T}_i) \frac{\mu_{\mathcal{T}_i(k)}}{\sigma^2_{\mathcal{T}_i(k)}}}{\frac{1}{\rho^2_{\mathbf{u}}} + \sum_{i,k|\mathcal{S}_i(k)=\mathbf{u}} q(\mathcal{T}_i) \frac{1}{\sigma^2_{\mathcal{T}_i(k)}}}, \tag{6.14}$$

which balances the votes from different epitome patches with the observed value for the pixel based on the ratio of appropriate noise or uncertainty parameters (variances σ^2 for epitome 'votes' and ρ^2 for the votes from the observed image), as well as the uncertainties

[5]Strictly speaking, the model is not normalized because of the factors g, but the same inference procedures can still be used; the imbalance due to g factors is uniformly distributed over the data, due to the fixed relative configuration of the neighborhoods N_i

about mapping $q(\mathcal{T}_i)$. The other update rules (6.7) remain the same, except that instead of patches $z_{\mathcal{S}_i}$, patches of variational hidden image means $\nu_{\mathcal{S}_i}$ are used to compute h_i.

We have performed a number of experiments on halucinating plausible guesses for large chunks of missing data, by setting variances $\rho^2_{\mathbf{u}}$ for the missing data to high values. For instance, in one of the experiments, the data x is assumed to be a video of a hallway walk-through (with all the motion due to the cameraman's walking), but only a *single* frame is given. In this case, the coordinates $\mathbf{u} = (x, y, t)$ are 3D, the patches $\mathcal{S}_i$ are all video cubes of a certain size, and the variances $\rho^2_{x,y,t}$ are set to a high value everywhere except when $t = 0$, where it is set to a small value, thus overpowering the epitome predictions. For the epitome e we simply used a sequence of a walk-through of another hallway to be its mean, and set the epitome variances to a same small value everywhere (training the epitome on a larger number of such sequences would probably lead to better results), and then iteratively applied equations (6.7, 6.14) until convergence. After each application of these equations, the inferred video ν resembles the original video, which we used as an epitome, more and more, both in terms of the local video texture resulting from quilting patches $e_{\mathcal{T}}$ and in terms of how the quilting of such patches in one part of the video volume influences the choice of the patches in another, distant, part of the volume. Thus, the resulting sequence $\nu_{x,y,t}$ contains the given photograph as its frame 0, since the low variances $\rho^2_{x,y,t=0}$ require it, but from $t = -7$ to $t = 7$ it adds new frames that agree with frame 0 so that the sequence contains the motion of the hall's walls out of the field of view, zooming motion of the texture close to the center of the field of view, as well as the same rocking motion of the human walk, present in the epitomic example.

In another experiment, the data and epitome coordinates are x, y, θ, where θ is a illumination angle, and the same procedure is used to perform single-example photograph relighting. Due to complex long-range correlations in these two types of data, inference of missing data using traditional patch quilting would be impossible.

6.5 Experiments

6.5.1 Epitome learning

In the original image epitome, a variety of patch sizes could be used during learning. Using large patches it is possible to capture large features, but because larger image structures also undergo larger deformations, the use of large patches also introduces significant amount of blurring. Small patches, on the other hand can capture repeating details that are easier to register, but the epitomes tend to have smaller structures and more discontinuities. When learning epitomes with long-range patch correlations, it is possible to capture large image structures using smaller patches, and thus achieve sharper epitomes and higher epitome likelihoods. The large epitome structures are the result of a combination of the global correlations provided by the mapping field we introduce in this chapter, and the local correlations provided simply by patch overlaps, as in the original epitome.

The epitome shown in Fig. 6.1 was learnt from just three images of cars. The mean of the three images was used to initialize the epitome and after learning, the resulting epitome is a morph of the three cars. No alignment of the images was done beforehand. The long-range patch correlations caused the patches from these three cars to essentially agree upon an alignment of their features. The car images and the epitome all have a resolution of 120x90 pixels, and patches of size 10x10 with 10 random correlation links were used during learning. An example of these patch correlation links is outlined on top of the image on the left. The patch, $\mathcal{S}_i$ is randomly linked to a couple other patches, such that their corresponding epitome mappings $\mathcal{T}_i$ keep a similar spatial configuration.

Fig. 6.3 illustrates the impact of long-range patch correlations in learning the epitome. The face epitome from [25] is shown on the left. Continuing the learning procedure for just a few more iterations, but including the constraints g on patch mappings by iterating (6.7, 6.10), results in the epitome on the right. The contiguous features in the

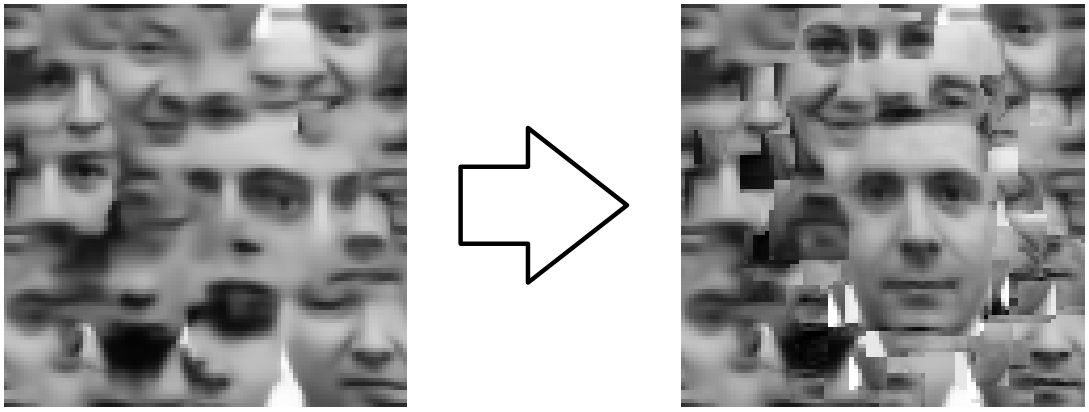

Figure 6.3: The effect of long-range patch correlations in learning the epitome. Starting with a traditional epitome on the left, conducting a few iterations of learning with patch correlations leads to the epitome on the right, which starts to show larger image structures, including a sharp whole prototypical face to which many of the images are mapped.

new epitome are significantly larger, with a prototypical sharp face emerging near the center that does not look like any one single face in the database.

We can also examine where patches in an image match in the epitome. With the patch correlation constraints, we expect patches of a human face to match to contiguous areas of the epitome, as opposed to patches scattered all around the epitome. But, when a non-human face is matched with the epitome, we expect constraints to be violated and patches would not match to the epitome in the same manner as would a human face. To show where patches in an image match in the epitome, the denominator in (6.10) can be used as a transparency mask on the epitome as shown in Fig. 6.4. The human face on the left causes a large contiguous area of the epitome to be used frequently. The large forehead of the subject also results in the high use of a bright patch above the main face area of the epitome. The middle image shows a digitally created image of a cyclops. The usage of the bottom half of the prototypical face in the epitome is normal,

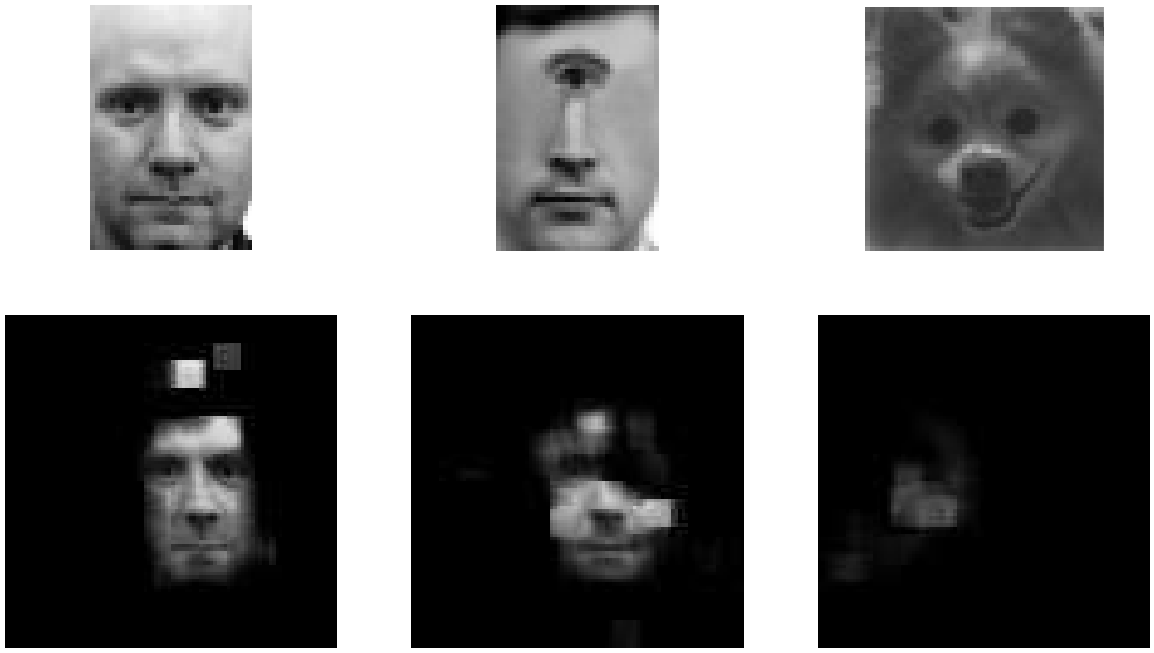

Figure 6.4: Face mapping. Three images of potential human faces are shown along the top with the corresponding matched areas of the epitome on the right from Fig. 6.3 below each one. The epitome is 'lit-up' proportional to how well the patches in each of the images matches to areas in the epitome.

but only one side of the upper half of the face is needed. Without modeling long-range correlations in epitome mappings, we would expect that both eyes would be used with equal probability, but because of these modeling constraints, for the most part, only half of the face in the epitome is used. Finally, an image of a dog is shown on the right. As it does not resemble a human face, the patch usage area in the epitome is quite deviant from that of a human face. Classification can be performed by computing the likelihood under the epitome for each of these images and the images shown have been ordered according to their likelihood with the human face on the left with the highest likelihood of the three.

6.5.2 Image illumination manipulation

It is often desired to change the illumination of a subject in order for it to appear consistent with other elements of a differently illuminated scene. Information from a training sequence can be leveraged and used to interpolate changes in illumination of an image. Fig. 6.5 shows an example. The top row shows several frames of a video sequence exhibiting a change in illumination. The single test image shown in the middle is then extrapolated to mimic the illumination change of the training sequence and the frames corresponding to those in the training set are shown.

The illumination change is transferred onto the image through patch matching between the image and the video sequence and subsequent transferral of the illumination change that the patches exhibit in the adjacent frames of the training sequence via the 3D nature of the patches. The result can then grow outwards in an iterative fashion. Because this is an extrapolation from a single image, it is difficult, especially in frames far from the original seed, to maintain the coherence of the patch matching. Using long-range correlations between the patches is essential in maintaining consistency in the results. The shadows in Fig. 6.5 move in a plausible fashion. Patches of size 10x10x5 were used with 30 correlation links.

The second face illumination example shown in Fig. 6.6 shows a wider range of illumination change over a different subject. The first row of results serves to demonstrate the need for the long-range correlations as that is the result without the correlation links, while the sequence in the bottom row incorporate such links.

In the final illumination experiment shown in Fig. 6.7, an analogous operation was performed with a rippled piece of clothing. The geometry of folded cloth is very complex and would be difficult to model. Again, the illumination change is transferred from a sample video sequence in order to extrapolate the change of illumination angle of the source lighting for a single image. The complexity of the subject posed a difficult problem, but even then, the shadows can be seen moving in a plausible manner.

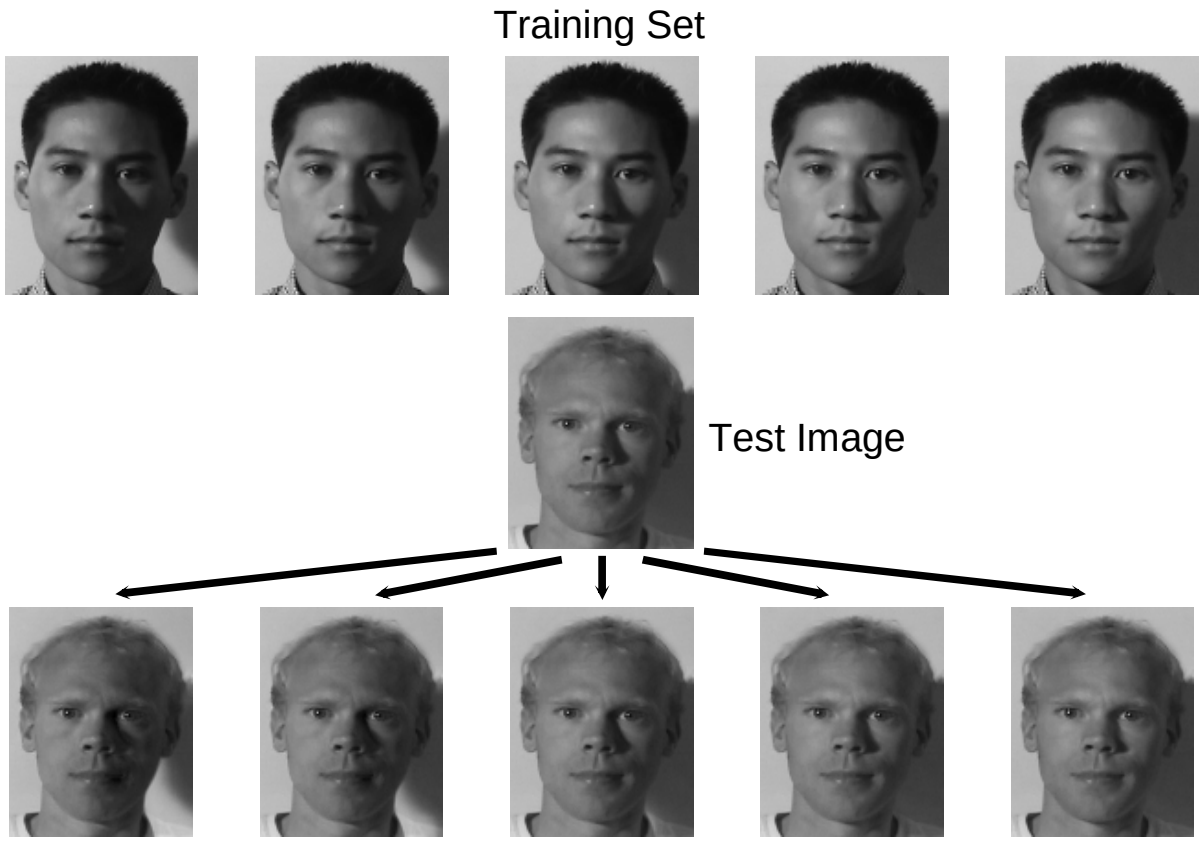

Figure 6.5: Changing the illumination of a face. Given a single test image and a guiding training set, a synthesized sequence is generated that reflects a changing illumination of the single image by iterating (6.7, 6.8, 6.9, 6.10, 6.14). The synthesis is plausible despite the absence of the use of any geometry or domain knowledge. Not only do the sharp shadows on the face move as expected, but the projected shadow behind the head also moves in a plausible manner. The training video of size 105x130 was used to synthesize 28 frames of size 100x125 from the target frame using patches of size 10x10x5 with 30 correlation links. The video sequence is available at http://www.psi.toronto.edu/∼vincent/patchcorr.html.

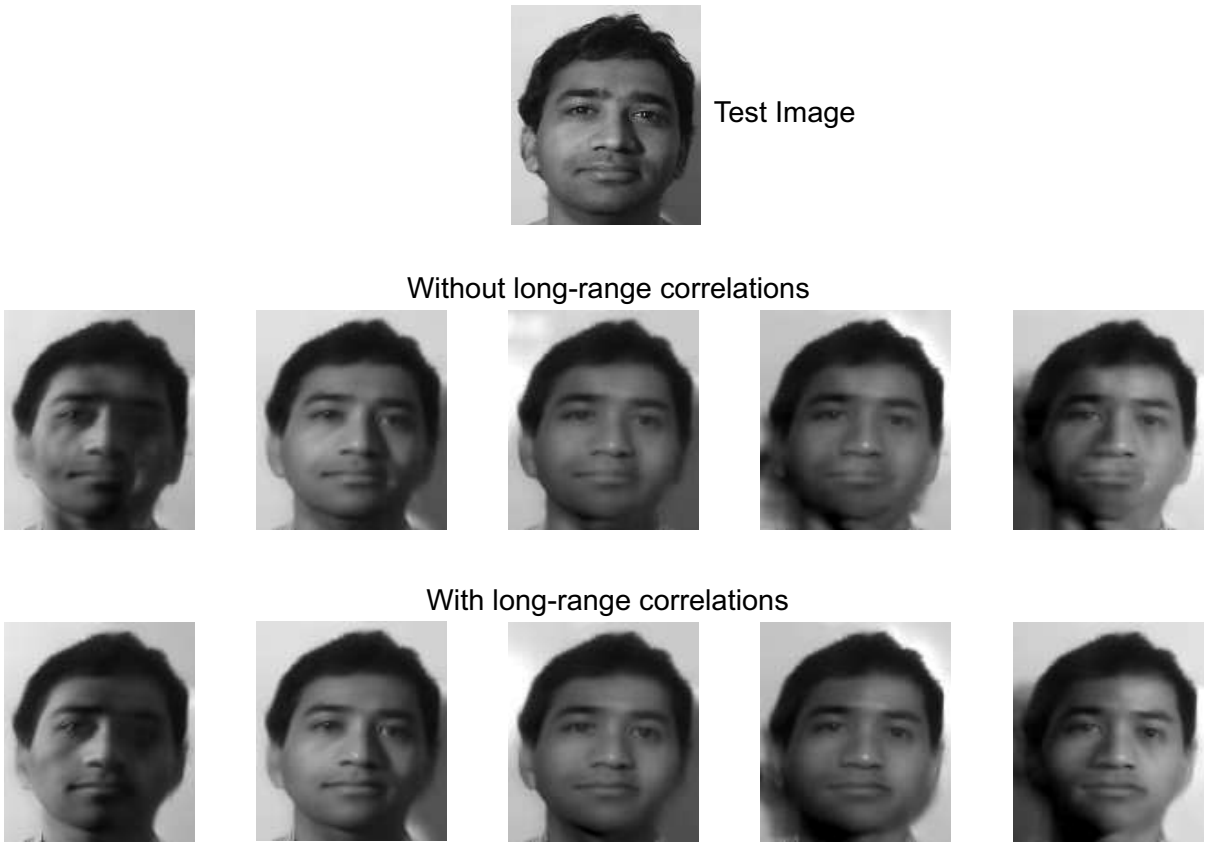

Figure 6.6: The necessity for long-range correlations in patch matching. The same experiment done in Fig. 6.5 is performed here with a different test image. Synthesis results with and without long-range patch correlations are shown. See the website for the video.

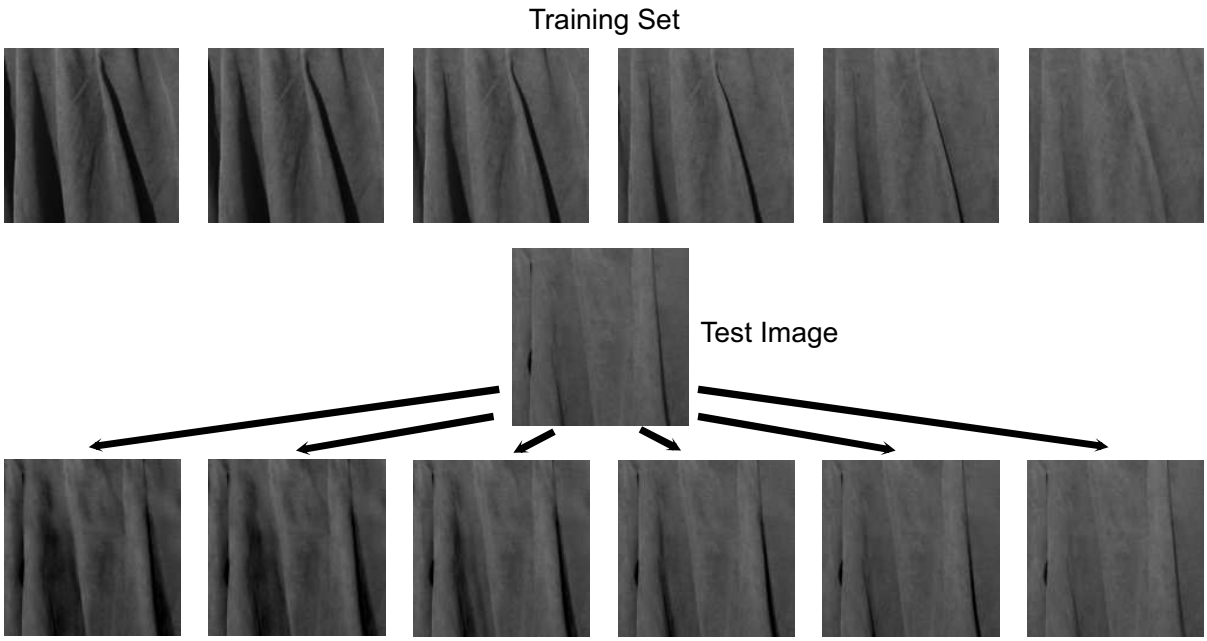

Figure 6.7: Changing the illumination of cloth. Given a single test image of draped cloth and a guiding sequence, the illumination of the single image is changed. Both the training set and the test image were of size 150x150, from which 74 frames were extrapolated using patches of size 15x15x5 with 50 correlation links. The video result can be found on the website.

6.5.3 Image walk-through

The same algorithm can be used in a variety of other synthesis applications. In Fig. 6.8, walking through a given image of a hallway is simulated given a training video. Note that the image is not simply enlarged, as parallax effects are apparent. As with the previous applications, no knowledge of geometry or domain knowledge is given to the algorithm. The patch correlations are sufficient to generate a plausible synthesis. The synthesis is blurry, but shows evidence of the transfer of the motion pattern. The short amount of training data and the extreme differences in the appearance of the hallway in the training video and test image lead to many patches being averaged that do not agree in their pixel values, leading to a blurry result. Additional training data and incorporating invariances beyond translation and illumination would improve the result. Nonetheless, the long-range correlations still guides the model to achieve a promising result.

6.6 Conclusion

We have introduced a powerful new patch-based model that accounts for the varying geometric configurations of object features to describe learnable probability density functions of visual data. The representation power of our model has been illustrated in a variety of tasks including multiple object registration and detection, as well as extreme missing data problems, such as relighting and walking through an image, where a single image frame is extrapolated to a video sequence, the video results of which can be found on the project webpage[6]. These tasks can be achieved without explicitly incorporating domain knowledge because our simple data-driven model captures sufficiently short- and long-range correlations among the data patches.

[6]http://www.psi.toronto.edu/∼vincent/patchcorr.html

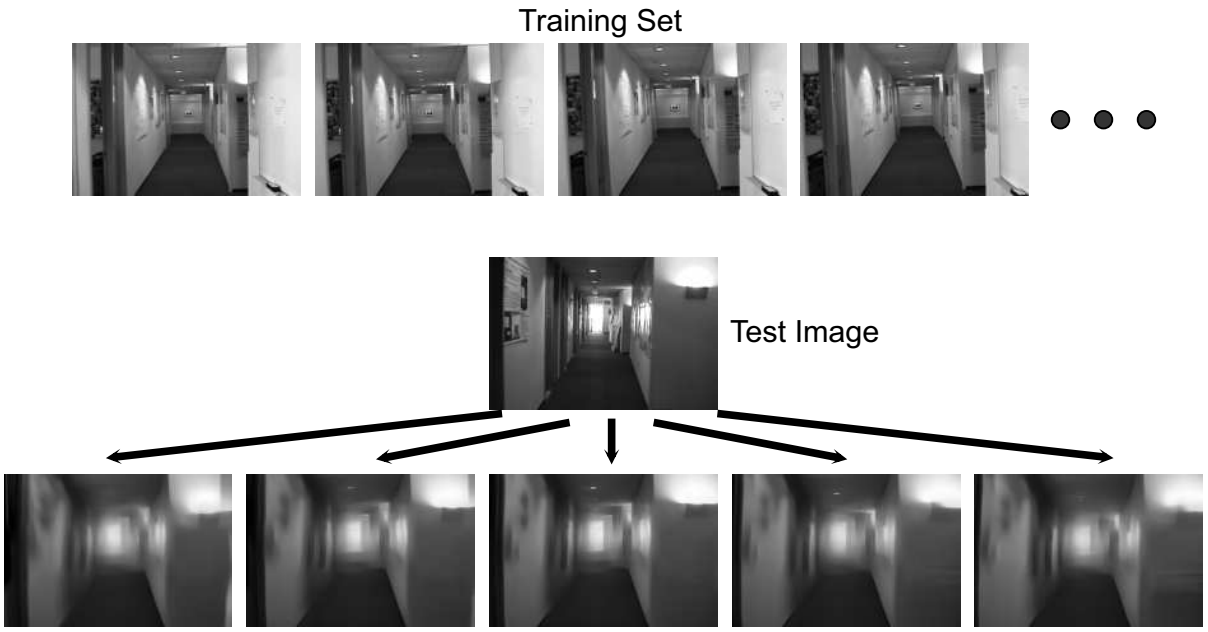

Figure 6.8: Walking through an image. Given a single image of a hallway, it is desired to mimic walking through the scene. Instead of just enlarging the image, it should appear as if the camera is moving down the hallway. This effect is achieved by quilting patches from a training sequence utilizing correlation links between patches to aid in matching. Patches of size 5x5x3 with 20 random correlation links were used to synthesize a plausible movement of walls, lights, and fixtures given the single 180x120 seed frame. See the video on the website for the full effect.

Chapter 7

Conclusion

This thesis has introduced two broadly applicable, principled, and efficient patch-based models for data processing applications. The power of these models have been demonstrated across a number of varied applications even without the use of any domain specific knowledge. Why these models work at all is because of their principled probability models.

The epitome model is a patch-based probability model that is learned by compiling together a large number of examples of patches from input data, and has a variety of data processing applications. This thesis has extended the epitome model in several ways to allow it to be used as a patch model platform for analyzing visual data and performing various tasks with the data.

The epitome size and patch size used in the epitome model was studied and by examining natural image statistics between the epitome and the original image, some guidance can be provided as to the appropriate values for these parameters. This technique works because the goal of the epitome model is to contain the important features of the original image. This work could be extended by using these natural image statistics to automatically determine an appropriate epitome and patch size for a given image. The size of the epitome could also be grown dynamically during the learning procedure based on its

ability to model the statistics of the original image.

Of practical concern to using the epitome model is computational complexity, as many patch comparisons are required to be performed during both learning and inference. A novel algorithm, the shifted cumulative sum algorithm, was introduced that scales better and is an order of magnitude faster than other known techniques. Particularly powerful is its ability to accommodate multiple patch sizes and shapes simultaneously. Further, the ability to execute this algorithm in parallel allows it to take advantage of multi-core processors and distributed cloud computing. With the advance in graphics processing units, future work adapting the patch comparison algorithm to run on these computing units could further improve performance.

The shifted cumulative sum algorithm makes it practical to extend the epitome model beyond 2D patches. Video epitomes were introduced as an extension to the basic image epitome model by adding 'time' as the third dimension to the patches from a video. The model was extended further to allow the video epitome to be trained directly on corrupted or degraded data, whereby the missing information would be filled in through repetitions in the data. Applications of video epitomes for video super-resolution, video interpolation, object removal, and denoising were demonstrated. The epitome model could be generalized further to incorporate other data besides pixel data such as descriptors in order to provide invariances such as rotation, scale, colour, and lighting in order to increase its modeling power.

Finally, this thesis introduced a powerful new patch-based model that accounts for the varying geometric configurations of object features to describe learnable probability density functions of visual data. This power of this model, when integrated with the epitome model, was demonstrated on a variety of tasks including multiple object registration and detection, as well as extreme missing data problems, such as relighting and walking through an image, where a single image frame is extrapolated to a video sequence. While all these applications of the models presented in this thesis are impressive on their own,

even more impressive is that these tasks can be achieved without explicitly incorporating domain knowledge because these simple data-driven models capture sufficiently short- and long-range correlations among the data patches. Incorporating this powerful patch-based model in the epitome proved fruitful and future work would include adapting the model to work within other patch-based models.

Bibliography

[1] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *Intern. Journal of Computer Vision*, pages 283–312, 1989.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[3] M. Ashikhmin. Synthesizing natural textures. In *Proc. Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

[4] J. Barros, J. French, W. Martin, P. Kelly, and M. Cannon. Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval. In *Proc. SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases*, volume IV, pages 392–403, 1996.

[5] J. L. Bentley. Multidimensional binary search trees used for associated searching. *Communications of the ACM*, 18(9):509–517, 1975.

[6] C. Bishop, A. Blake, and B. Marthi. Super-resolution enhancement of video. In *Proc. Artificial Intelligence and Statistics*, 2003.

[7] C. Broit. *Optimal registration of deformed images*. PhD thesis, University of Pennsylvania, 1981.

[8] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Proc. IEEE Comp. Vision and Pattern Recognition*, pages 60–65, 2005.

[9] M. C. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *Proc. ECCV*, pages 628–641, 1998.

[10] V. Cheung, B. J. Frey, and N. Jojic. Video epitomes. In *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*, pages 42–49, 2005.

[11] A. Criminisi, P. Pérez, and K. Toyama. Object removal by exemplar-based inpainting. In *Proc. Conf. on Computer Vision and Pattern Recognition*, pages 721–728, 2003.

[12] P. Debevec, A. Wenger, C. Tchou, A. Gardner, J. Waese, and T. Hawkins. A lighting reproduction approach to live-action compositing. In *SIGGRAPH*, pages 547–556, 2002.

[13] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proc. SIGGRAPH*, pages 341–346, 2001.

[14] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. Int. Conf. on Comp. Vision*, pages 1033–1038, 1999.

[15] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*, page 264, 2003.

[16] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, pages 56–65, 2002.

[17] B. Frey and N. Jojic. Advances in algorithms for inference and learning in complex probability models. *Accepted, IEEE Trans. PAMI*, pages 12–36, 2003.

[18] B. J. Frey and N. Jojic. A comparison of algorithms for inference and learning in probabilistic graphical models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(9):1392–1416, 2005.

[19] A. Fusiello, V. Roberto, and E. Trucco. Efficient stereo with multiple windowing. In *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*, pages 858–863, 1997.

[20] A. Fusiello, V. Roberto, and E. Trucco. Efficient stereo with multiple windowing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 858–863, 1997.

[21] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proc. SIGGRAPH*, pages 327–340, 2001.

[22] J. Huang and D. Mumford. Statistics of natural images and models. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*, 1999.

[23] M. Isard and J. MacCormick. Bramble: A bayesian multiple-blob tracker. In *Proc. Int. Conf. on Comp. Vision*, pages 34–41, 2001.

[24] A. Jepson and M. Black. Mixture models for optical flow computation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 760–761, 1993.

[25] N. Jojic, B. J. Frey, and A. Kannan. Epitomic analysis of appearance and shape. In *Proc. IEEE Intern. Conf. on Computer Vision*, pages 34–41, 2003.

[26] N. Jojic, V. Jojic, B. J. Frey, C. Meek, and D. Heckerman. Using 'epitomes' to model genetic diversity: Rational design of hiv vaccine cocktails. In *Proc. NIPS*, 2005.

[27] A. Kannan, J. Winn, and C. Rother. Clustering appearance and shape by learning jigsaws. In *Proc. NIPS*, 2006.

[28] A. Kapoor and S. Basu. The audio epitome: a new representation for modeling and classifying auditory phenomena. In *Proc. ICASSP*, 2005.

[29] W. Kim and J.M. Rehg. Detection of unnatural movement using epitomic analysis. In *Proc. ICMLA*, pages 271–276, 2008.

[30] F. F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*, pages 524–531, 2005.

[31] L. Liang, C. Liu, Y. Q. Xu, B. Guo, and H. Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20(3):127–150, 2001.

[32] R. M. Neal and G. E. Hinton. A new view of the em algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. 1998.

[33] A. Nealen and M. Alexa. Hybrid texture synthesis. In *Proc. Eurographics Symposium on Rendering*, pages 97–105, 2003.

[34] R. Rosales, K. Achan, and B. J. Frey. Unsupervised image translation. In *Proc. IEEE Intern. Conf. on Computer Vision*, pages 472–478, 2003.

[35] D.L. Ruderman. The statistics of natural images. *Network Computation in Neural Systems*, 5:517–548, 1994.

[36] A. Shashua and T. Riklin-Raviv. The quotient image: class-based re-rendering and recognition withvarying illuminations. *IEEE Trans. PAMI*, 23(2):129–139, 2001.

[37] M.J. Swain and D.H. Ballard. Color indexing. *Intern. J. Computer Vision*, pages 11–32, 1991.

[38] J. Y. A. Wang, E. H. Adelson, and U. Y. Desai. Applying mid-level vision techniques for video data compression and manipulation. In *Proc. SPIE on Digital Video Compression on Personal Computers: Algorithms and Technologies*, pages 116–127, 1994.

[39] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *Proc. ECCV*, pages 18–32, 2000.

[40] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In *Proc. IEEE Comp. Vision and Pattern Recognition*, pages 120–127, 2004.

[41] L. Zhang, S. Wang, and D. Samaras. Face synthesis and recognition under arbitrary unknown lighting using a spherical harmonic basis morphable model. In *Proc. IEEE CVPR*, pages 209–216, 2005.

[42] S. C. Zhu, C. Guo, Y. N. Wu, and Y. Wang. What are textons? In *Proc. of the 7th European Conf. on Computer Vision-Part IV*, pages 793–807, 2002.