Conditional Probability Models for Deep Image Compression – Suppl. Material

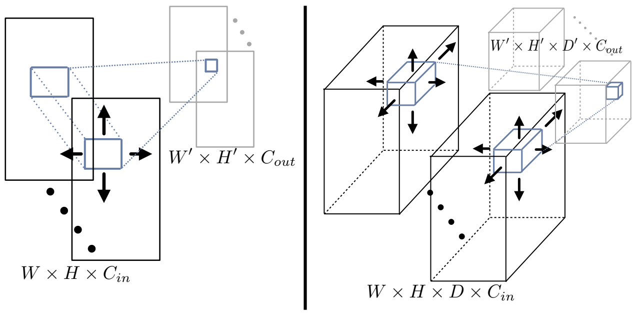

$W' \times H' \times C_{out}$
$W \times H \times C_{in}$
$W' \times H' \times D' \times C_{out}$
$W \times H \times D \times C_{in}$

Figure 7: 2D vs. 3D CNNs

3D probability classifier As mentioned in Section 3.2, we rely on masked 3D convolutions to enforce the causality constraint in our probability classifier P. In a 2D-CNN, standard 2D convolutions are used in filter banks, as shown in Fig. 7 on the left: A $W \times H \times C_{\text{in}}$-dimensional tensor is mapped to a $W' \times H' \times C_{\text{out}}$-dimensional tensor using C_{out} banks of C_{in} 2D filters, i.e., filters can be represented as $f_W \times f_H \times C_{\text{in}} \times C_{\text{out}}$-dimensional tensors. Note that all C_{in} channels are used together, which violates causality: When we encode, we proceed channel by channel.

Using 3D convolutions, a depth dimension D is introduced. In a 3D-CNN, $W \times H \times D \times C_{\text{in}}$-dimensional tensors are mapped to $W' \times H' \times D' \times C_{\text{out}}$-dimensional tensors, with $f_W \times f_H \times f_D \times C_{\text{in}} \times C_{\text{out}}$-dimensional filters. Thus, a 3D-CNN slides over the depth dimension, as shown in Fig. 7 on the right. We use such a 3D-CNN for P, where we use as input our $W \times H \times K$-dimensional feature map $\hat{\mathbf{z}}$, using $D = K, C_{\text{in}} = 1$ for the first layer.

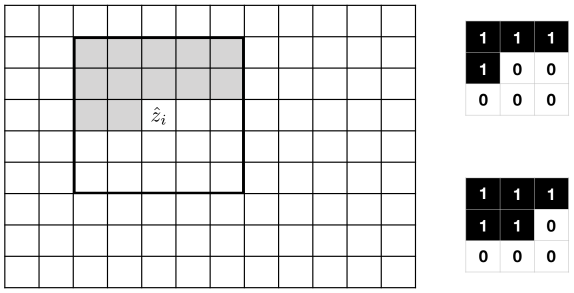


Figure 8: Left shows a grid of symbols $\hat{z}_i$, where the black square denotes some context and the gray cells denote symbols which where previously encoded. Right shows masks.

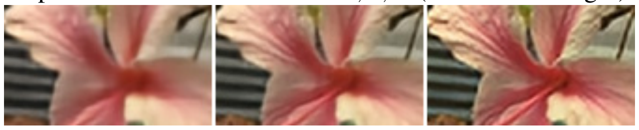
To explain how we mask the filters in P, consider the 2D case in Fig. 8. We want to encode all values $\hat{z}_i$ by iterating in raster scan order and by computing $p(\hat{z}_i | \hat{z}_{i-1}, \dots, \hat{z}_1)$. We simplify this by instead of relying on all previously encoded symbols, we use some $c \times c$-context around $\hat{z}_i$ (black square in Fig. 8). To satisfy the causality constraint, this context may only contain values above $\hat{z}_i$ or in the same row to the left of $\hat{z}_i$ (gray cells). By using the filter shown in Fig. 8 in the top right for the first layer of a CNN and the filter shown in Fig. 8 in the bottom right for subsequent filters, we can build a 2D-CNN with a $c \times c$ receptive field that forms such a context. We build our 3D-CNN P by generalizing this idea to 3D, where we construct the mask for the filter of the first layer as shown in pseudo-code Algorithm 1. The mask for the subsequent layers is constructed analoguously by replacing "$<$" in line 7 with "$\leq$". We use filter size $f_W = f_H = f_D = 3$.

Algorithm 1 Constructing 3D Masks

```
 1: $central\_idx \leftarrow \lceil (f_W \cdot f_H \cdot f_D)/2 \rceil$
 2: $current\_idx \leftarrow 1$
 3: $mask \leftarrow f_W \times f_H \times f_D$-dimensional matrix of zeros
 4: for $d \in \{1, \dots, f_D\}$ do
 5:     for $h \in \{1, \dots, f_H\}$ do
 6:         for $w \in \{1, \dots, f_W\}$ do
 7:             if $current\_idx < central\_idx$ then
 8:                 $mask(w, h, d) = 1$
 9:             else
10:                 $mask(w, h, d) = 0$
11:             $current\_idx \leftarrow current\_idx + 1$
```

With this approach, we obtain a 3D-CNN P which operates on $f_H \times f_W \times f_D$-dimensional blocks. We can use P to encode $\hat{\mathbf{z}}$ by iterating over $\hat{\mathbf{z}}$ in such blocks, exhausting first axis w, then axis h, and finally axis d (like in Algorithm 1). For each such block, P yields the probability distribution of the central symbol given the symbols in the block. Due to the construction of the masks, this probability distribution only depends on previously encoded symbols.

Multiple compression rates It is quite straightforward to obtain multiple operating points in a single network with our framework: We can simply share the network but use multiple importance maps. We did a simple experiment where we trained an autoencoder with 5 different importance maps. In each iteration, a random importance map i was picked, and the target entropy was set to $i/5 \cdot t$. While not tuned for performance, this already yielded a model competitive with BPG. The following shows the output of the model for $i = 1, 3, 5$ (from left to right):



On the benefit of 3DCNN and joint training We note that the points from Table 1 (where we trained different en-

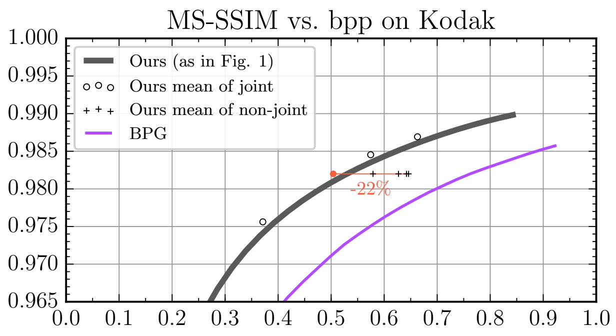


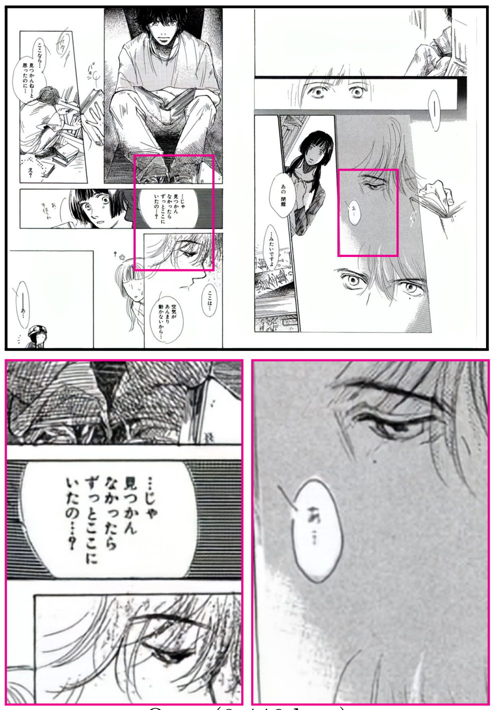
Figure 9: Performance on the Kodak dataset. See text.

tropy models non-jointly as a post-training step) are not directly comparable with the curve in Fig. 1. This is because these points are obtained by taking the mean of the MS-SSIM and bpp values over the Kodak images for a single model. In contrast, the curve in Fig. 1 is obtained by following the approach of [14], constructing a MS-SSIM vs. bpp curve per-image via interpolation (see *Comparison* in Section 4). In Fig. 9, we show the black curve from Fig. 1, as well as the mean (MS-SSIM, bpp) points achieved by the underlying models (∘). We also show the points from Tab. 1 (+). We can see that our masked 3DCNN with joint training gives a significant improvement over the separately trained 3DCNN, i.e., a 22% reduction in bpp when comparing mean points (the red point is estimated).

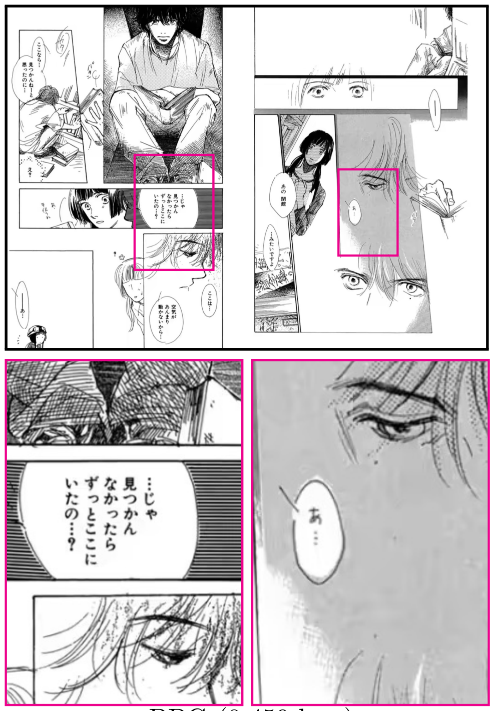
Non-realistic images In Fig. 10, we compare our approach to BPG on an image from the Manga109[8] dataset. We can see that our approach preserves text well enough to still be legible, but it is not as crip as BPG (left zoom). On the other hand, our approach manages to preserve the fine texture on the face better than BPG (right zoom).

Visual examples The following pages show the first four images of each of our validation sets compressed to low bitrate, together with outputs from BPG, JPEG2000 and JPEG compressed to similar bitrates. We ignored all header information for all considered methods when computing the bitrate (here and throughout the paper). We note that the only header our approach requires is the size of the image and an identifier, e.g., β, specifying the model.

Overall, our images look pleasant to the eye. We see cases of over-blurring in our outputs, where BPG manages to keep high frequencies due to its more local approach. An example is the fences in front of the windows in Fig. 14, top, or the text in Fig. 15, top. On the other hand, BPG tends to discard low-contrast high frequencies where our approach keeps them in the output, like in the door in Fig. 11, top, or in the hair in Fig. 12, bottom. This may be explained by



Ours (0.446 bpp)



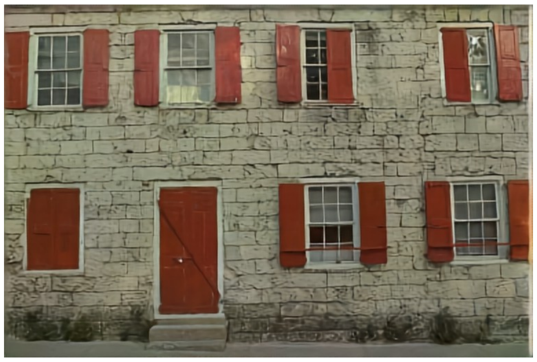
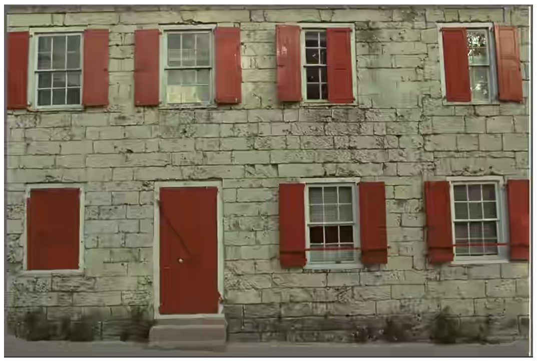
BPG (0.459 bpp)

Figure 10: Comparison on a non-realistic image. See text.

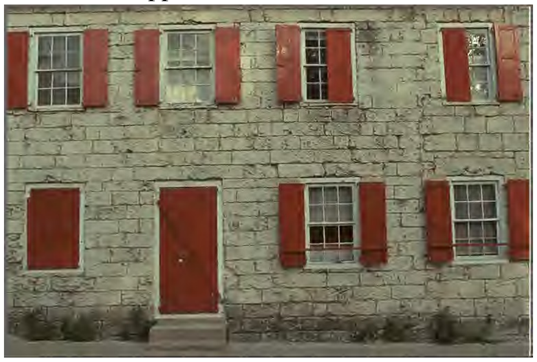
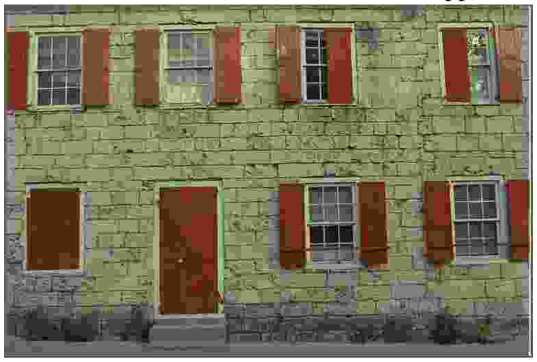
BPG being optimized for MSE as opposed to our approach being optimized for MS-SSIM.

JPEG looks extremely blocky for most images due to the very low bitrate.

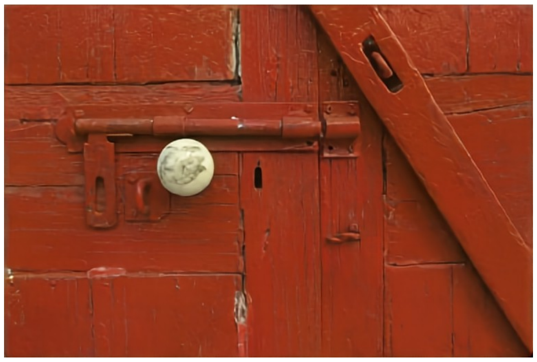
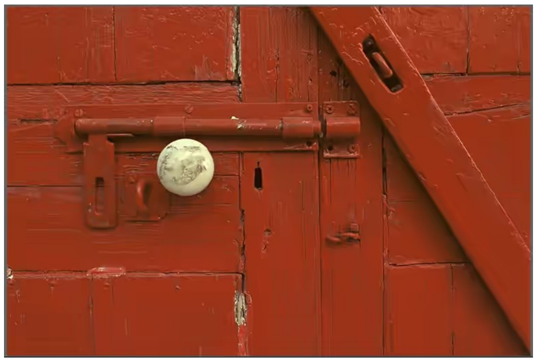
[8] http://www.manga109.org/

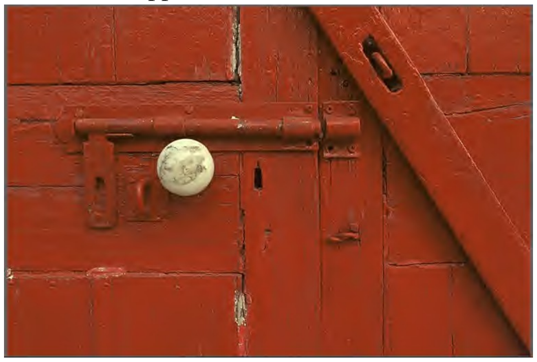
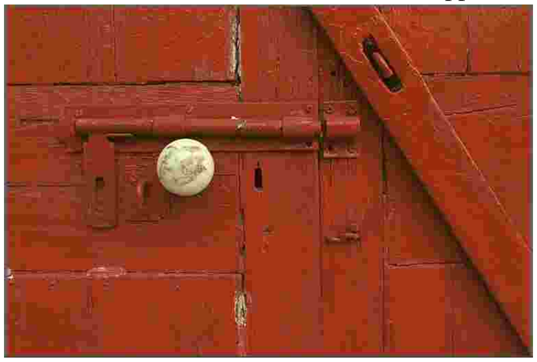
Ours 0.239 bpp 0.246 bpp **BPG**

JPEG 2000 0.242 bpp 0.259 bpp **JPEG**

Ours 0.203 bpp 0.201 bpp **BPG**

JPEG 2000 0.197 bpp 0.205 bpp **JPEG**

Figure 11: Our approach vs. BPG, JPEG and JPEG 2000 on the first and second image of the Kodak dataset, along with bit rate.

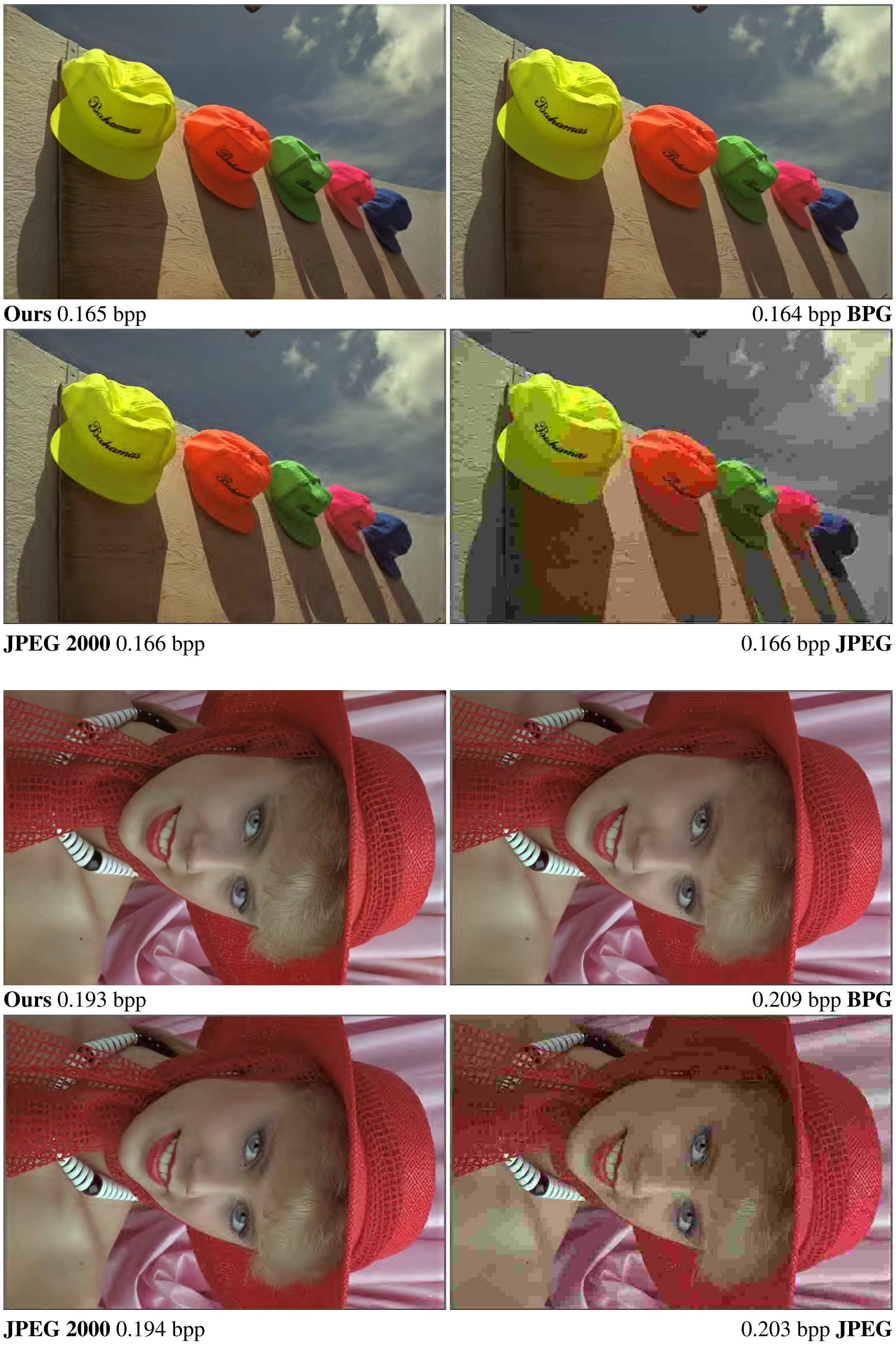
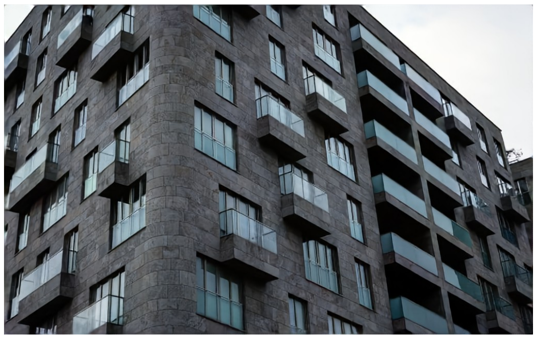
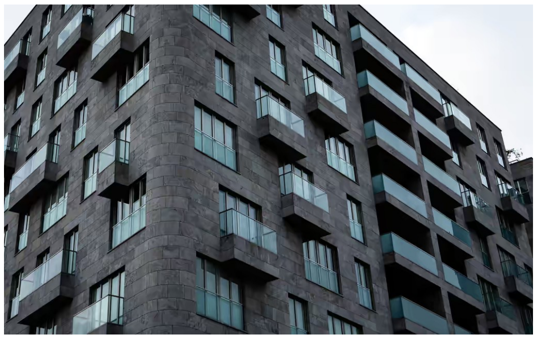
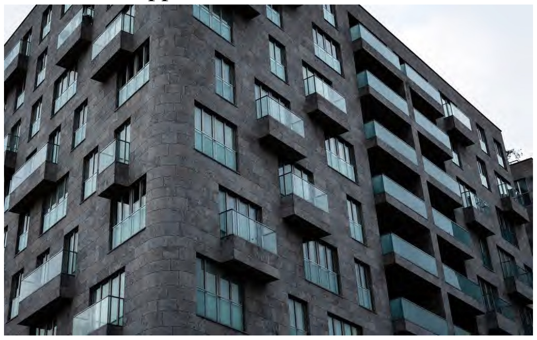
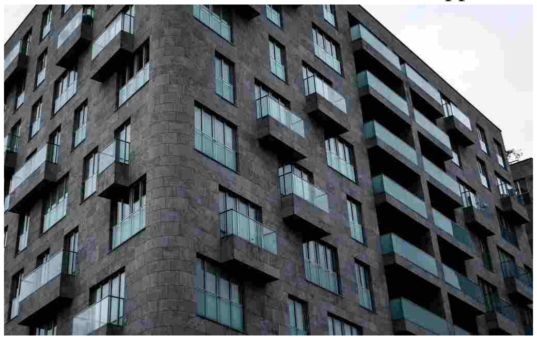


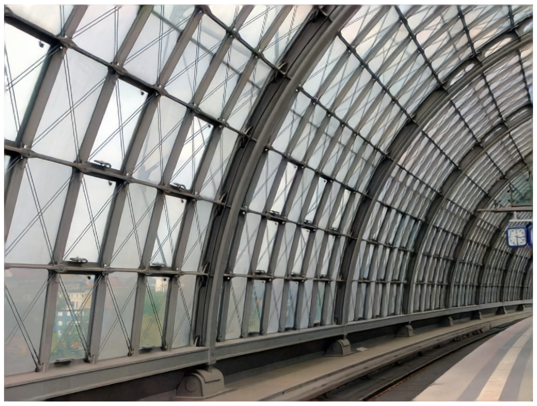
Figure 12: Our approach vs. BPG, JPEG and JPEG 2000 on the third and fourth image of the Kodak dataset, along with bit rate.

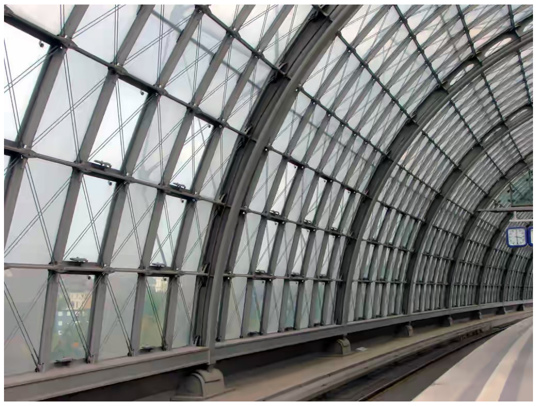

Ours 0.385 bpp

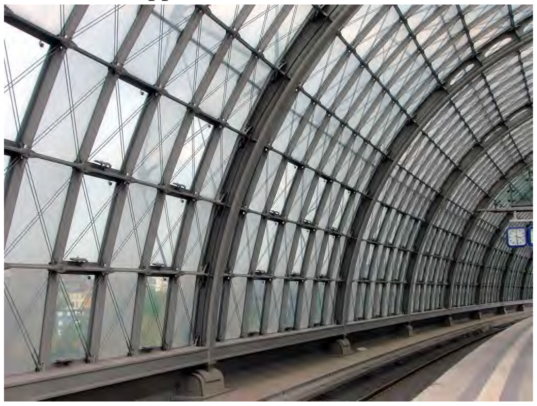

0.394 bpp **BPG**


JPEG 2000 0.377 bpp


0.386 bpp **JPEG**


Ours 0.365 bpp


0.363 bpp **BPG**


JPEG 2000 0.363 bpp

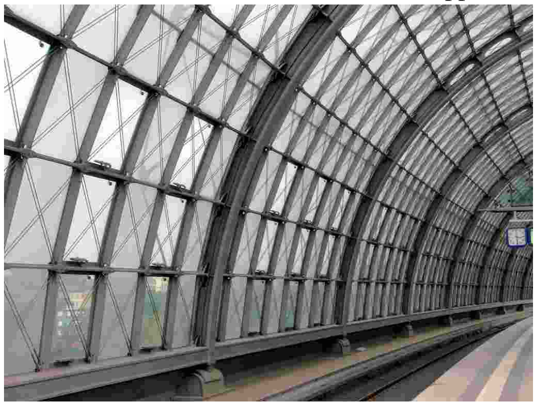
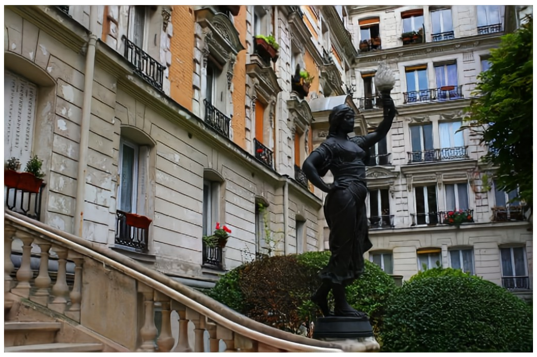
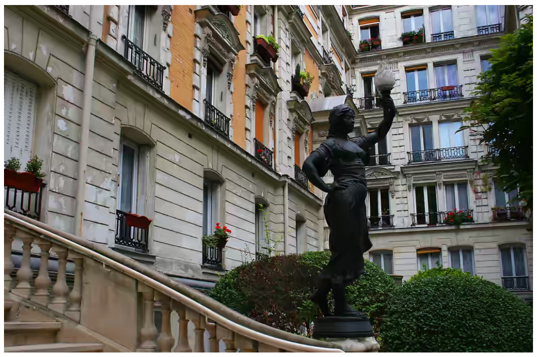
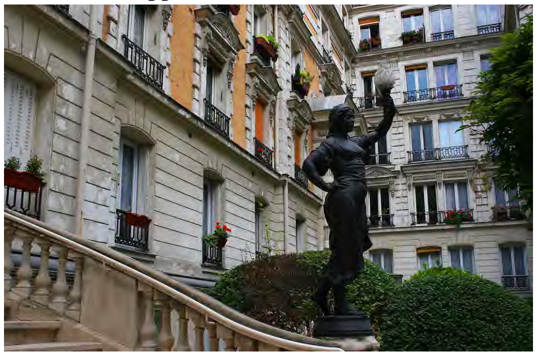
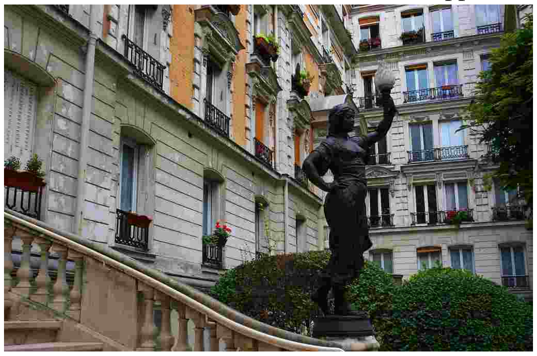

0.372 bpp **JPEG**

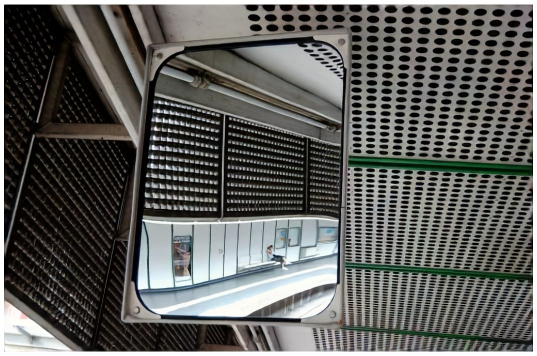
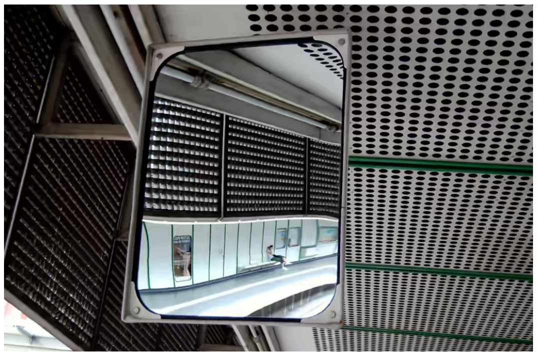
Figure 13: Our approach vs. BPG, JPEG and JPEG 2000 on the first and second image of the Urban100 dataset, along with bit rate.

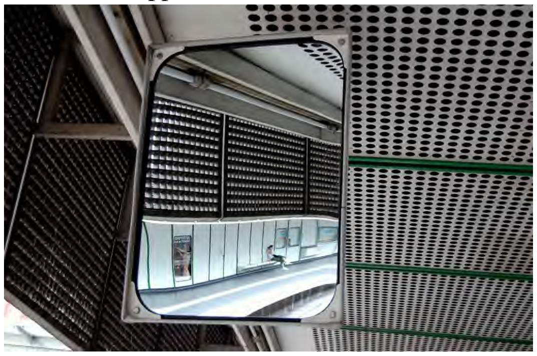
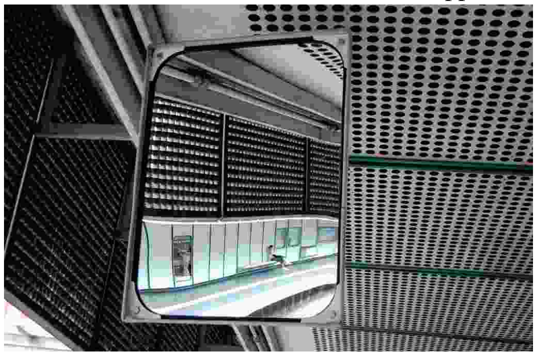
Ours 0.435 bpp 0.479 bpp **BPG**

JPEG 2000 0.437 bpp 0.445 bpp **JPEG**

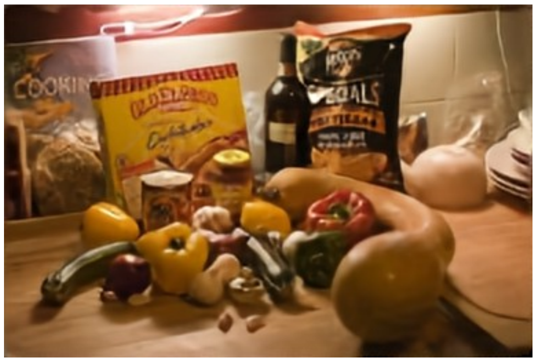
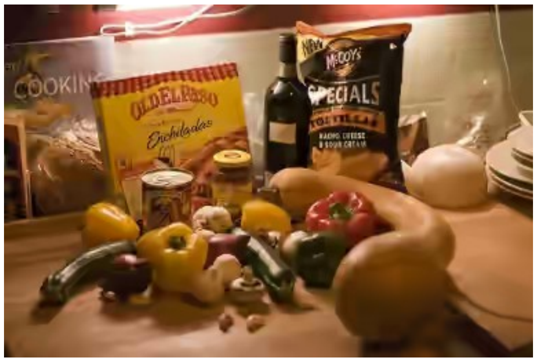



Ours 0.345 bpp 0.377 bpp **BPG**

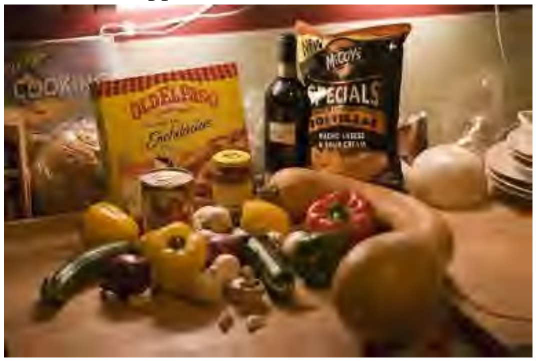
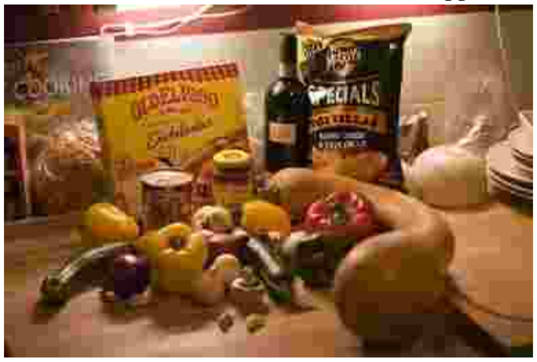



JPEG 2000 0.349 bpp 0.357 bpp **JPEG**

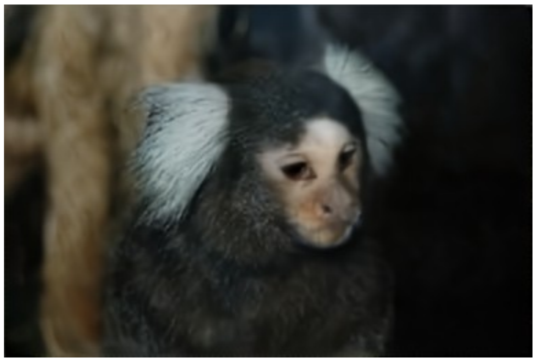
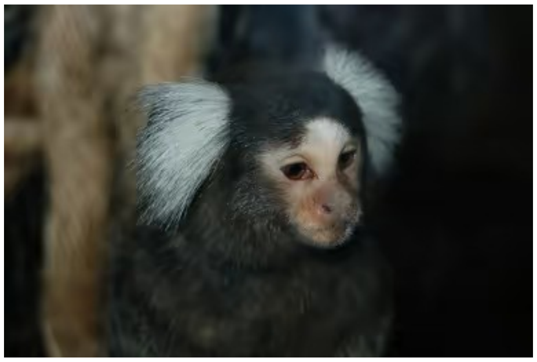
Figure 14: Our approach vs. BPG, JPEG and JPEG 2000 on the third and fourth image of the Urban100 dataset, along with bit rate.

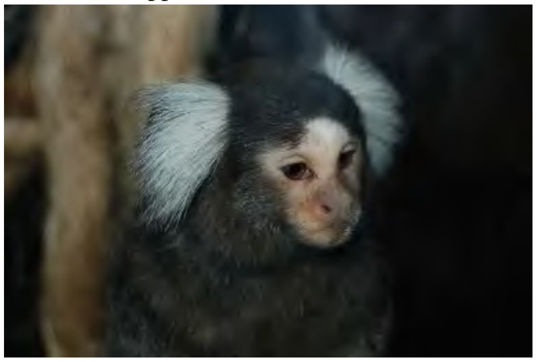
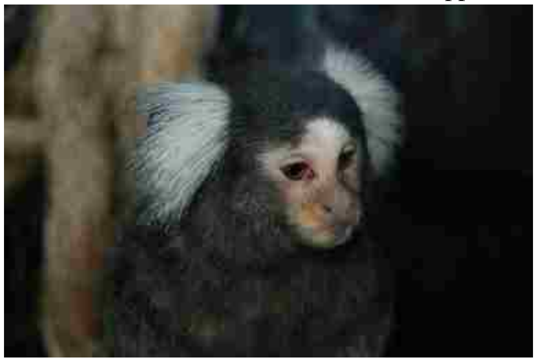
Ours 0.355 bpp 0.394 bpp **BPG**

JPEG 2000 0.349 bpp 0.378 bpp **JPEG**

Ours 0.263 bpp 0.267 bpp **BPG**

JPEG 2000 0.254 bpp 0.266 bpp **JPEG**

Figure 15: Our approach vs. BPG, JPEG and JPEG 2000 on the first and second image of the ImageNetTest dataset, along with bit rate.

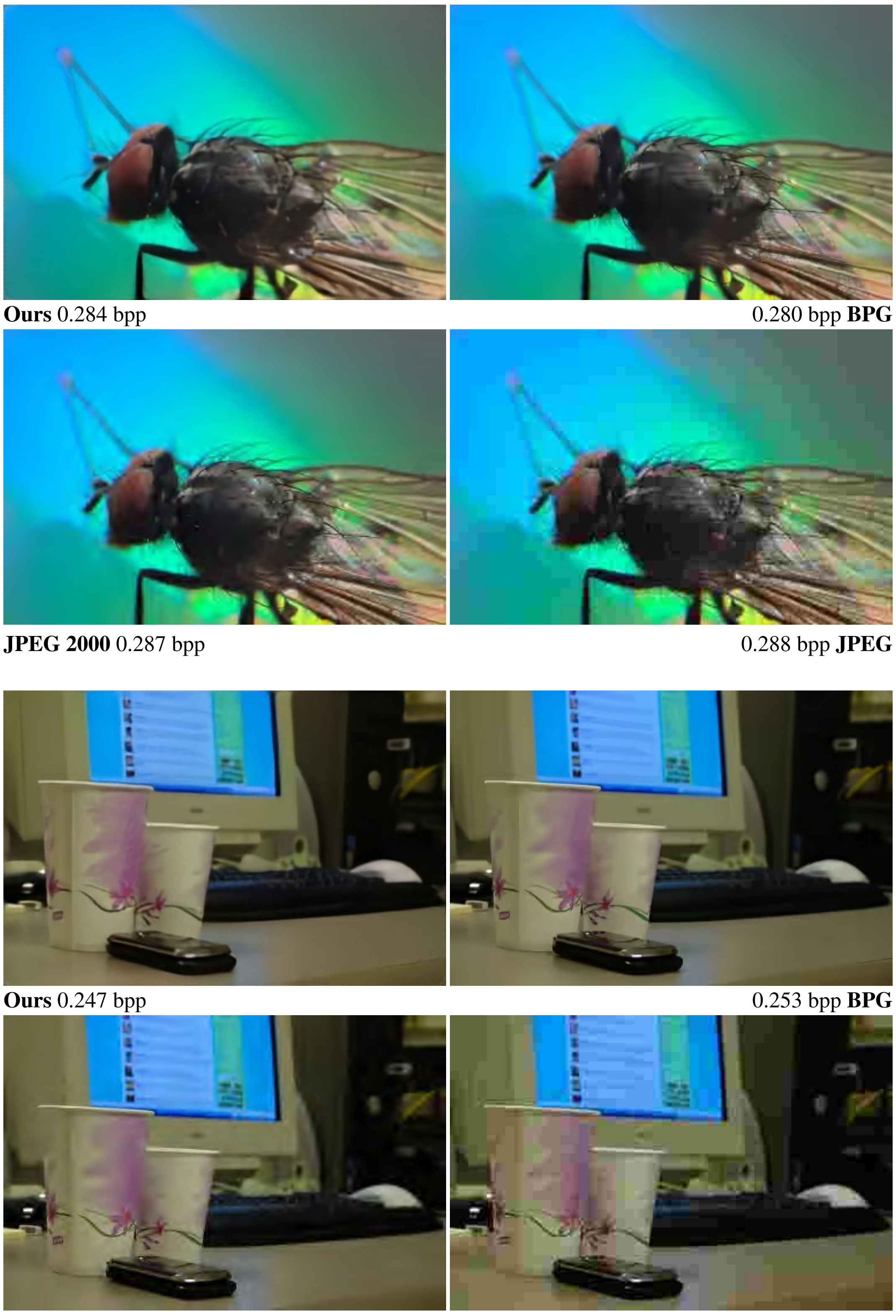
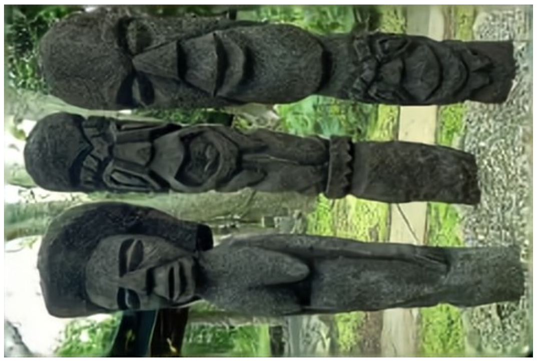
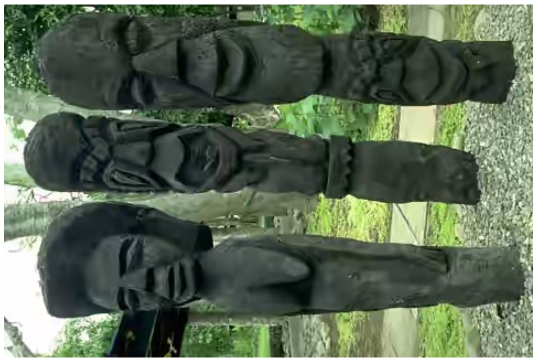
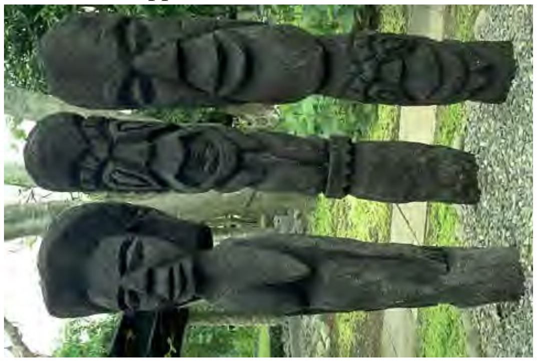
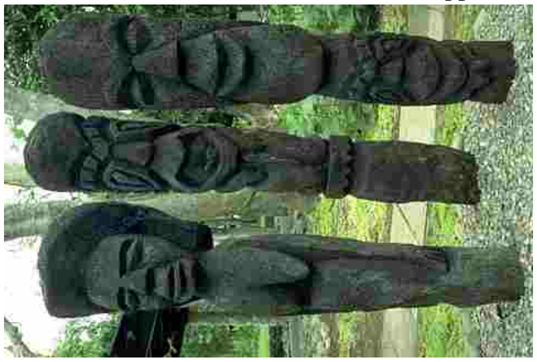


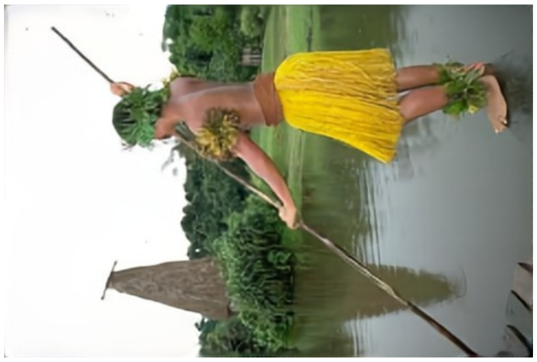
Figure 16: Our approach vs. BPG, JPEG and JPEG 2000 on the third and fourth image of the ImageNetTest dataset, along with bit rate.

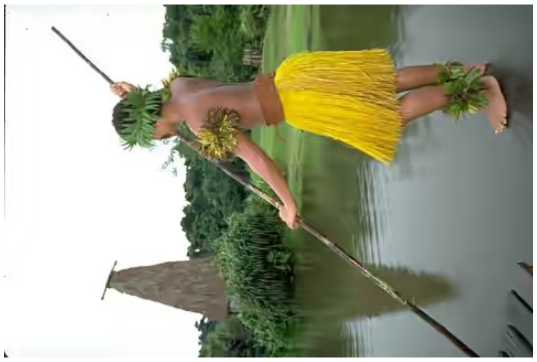

Ours 0.494 bpp

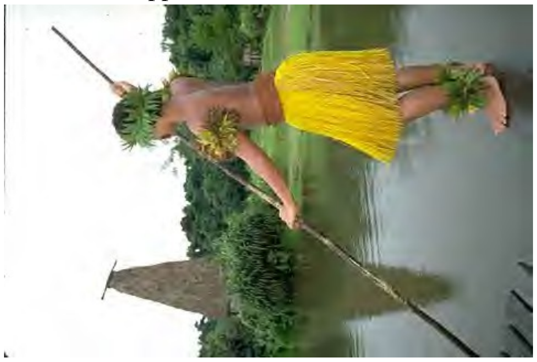

0.501 bpp **BPG**


JPEG 2000 0.490 bpp


0.525 bpp **JPEG**


Ours 0.298 bpp


0.301 bpp **BPG**


JPEG 2000 0.293 bpp

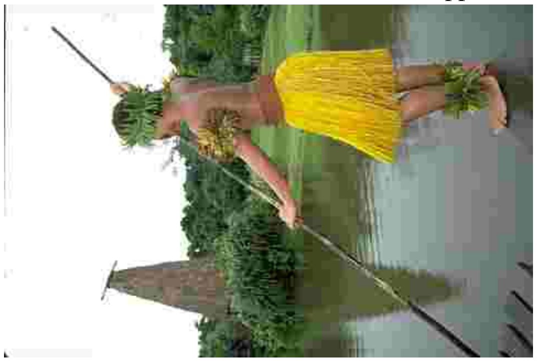
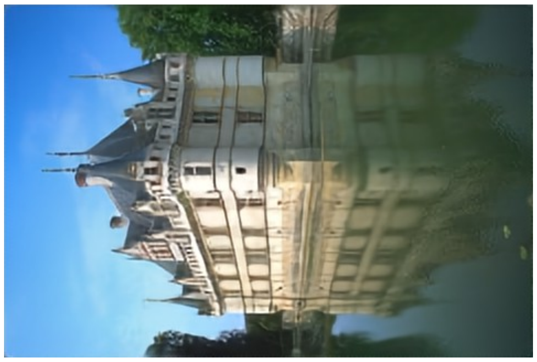
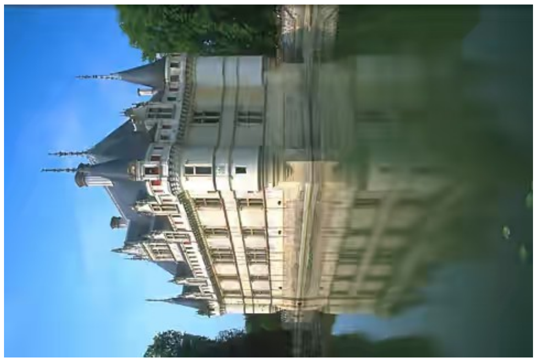
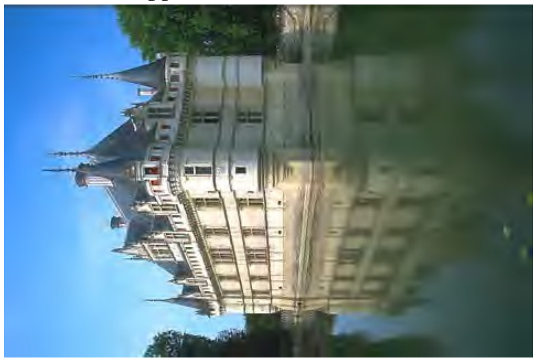
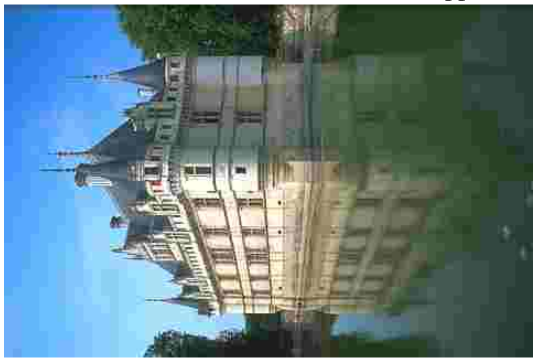

0.315 bpp **JPEG**

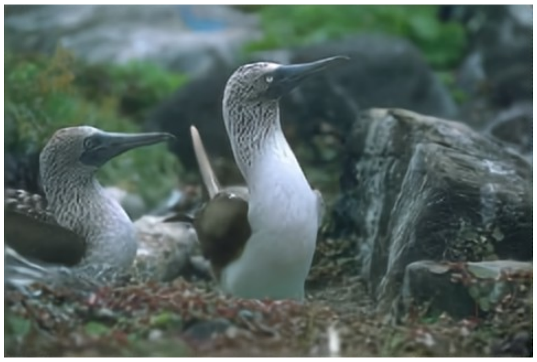
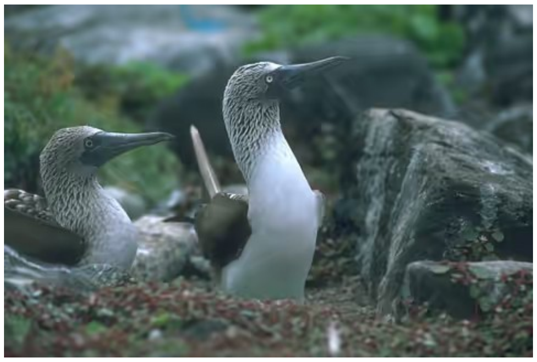
Figure 17: Our approach vs. BPG, JPEG and JPEG 2000 on the first and second image of the B100 dataset, along with bit rate.

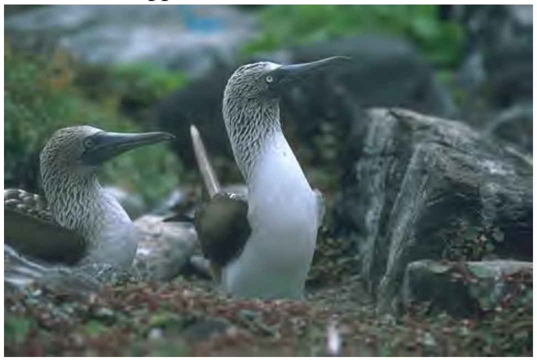
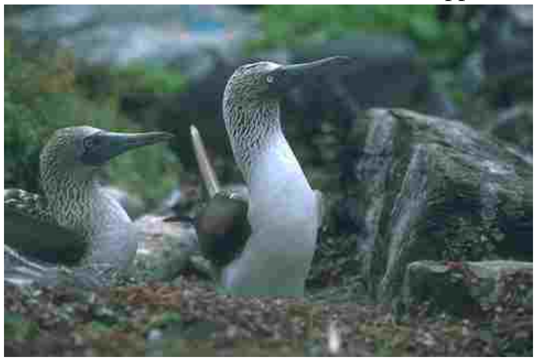
Ours 0.315 bpp 0.329 bpp **BPG**

JPEG 2000 0.311 bpp 0.321 bpp **JPEG**

Ours 0.363 bpp 0.397 bpp **BPG**

JPEG 2000 0.369 bpp 0.372 bpp **JPEG**

Figure 18: Our approach vs. BPG, JPEG and JPEG 2000 on the third and fourth image of the B100 dataset, along with bit rate.