Progressive Image Compression

CS6025 Data Encoding

Yizong Cheng

1-27-15

Sequential Decoding

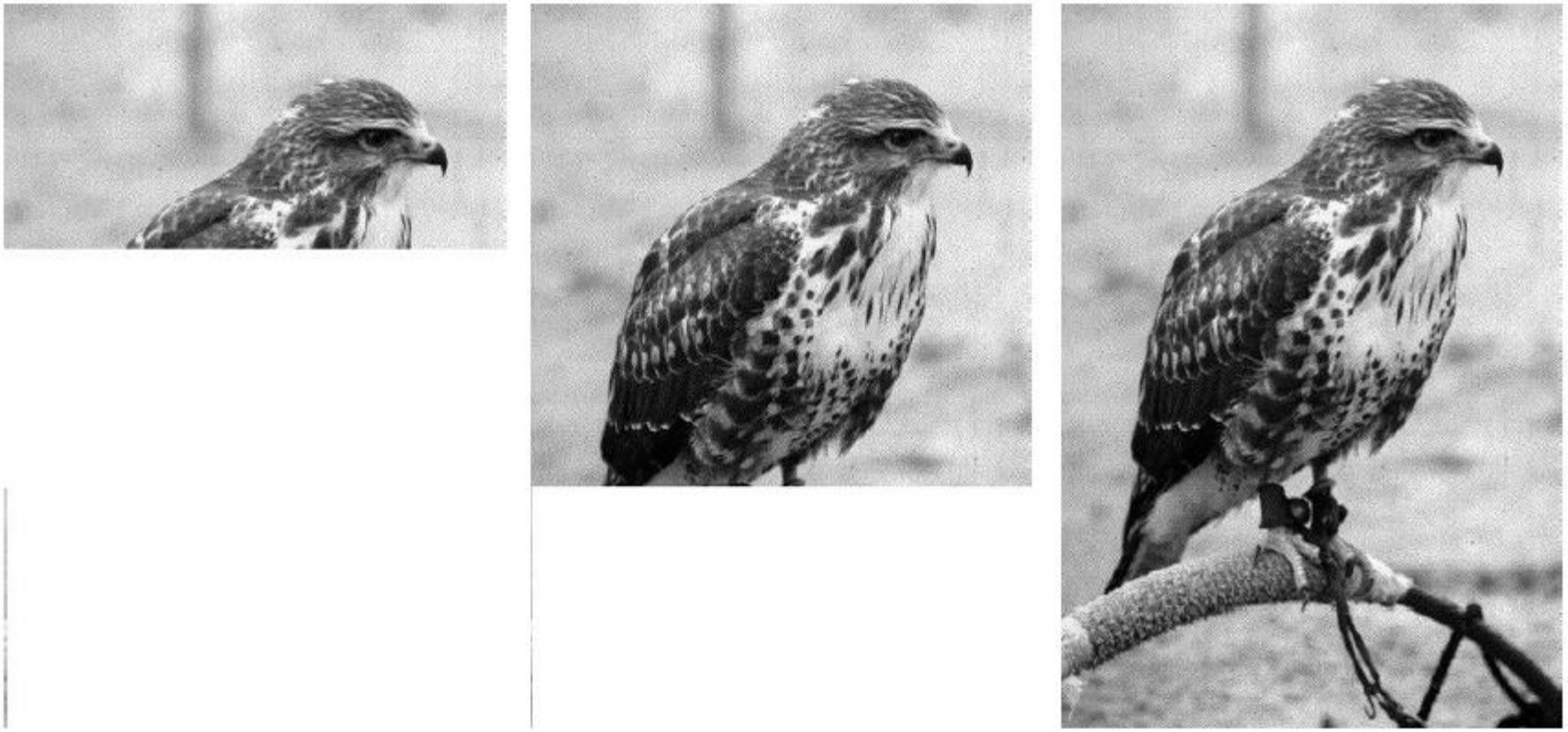


Figure 7.82: Sequential Decoding.

Progressive Decoding: Fuzzy to Crispy

Progressive Decoding in Shades

Progressive Decoding in Resolution

Layer 0: 1x1

0	16	4	19	1	22	7	25
17	18	20	21	23	24	26	27
5	28	6	31	8	34	9	37
29	30	32	33	35	36	38	39
2	40	10	43	3	46	13	49
41	42	44	45	47	48	50	51
11	52	12	55	14	58	15	61
53	54	56	57	59	60	62	63

Layer 1: 2x2

0	16	4	19	1	22	7	25
17	18	20	21	23	24	26	27
5	28	6	31	8	34	9	37
29	30	32	33	35	36	38	39
2	40	10	43	3	46	13	49
41	42	44	45	47	48	50	51
11	52	12	55	14	58	15	61
53	54	56	57	59	60	62	63

Layer 2: 4x4

0	16	4	19	1	22	7	25
17	18	20	21	23	24	26	27
5	28	6	31	8	34	9	37
29	30	32	33	35	36	38	39
2	40	10	43	3	46	13	49
41	42	44	45	47	48	50	51
11	52	12	55	14	58	15	61
53	54	56	57	59	60	62	63

Layer 3: 8x8

0	16	4	19	1	22	7	25
17	18	20	21	23	24	26	27
5	28	6	31	8	34	9	37
29	30	32	33	35	36	38	39
2	40	10	43	3	46	13	49
41	42	44	45	47	48	50	51
11	52	12	55	14	58	15	61
53	54	56	57	59	60	62	63

Layers

- Layer k for a 2^k x 2^k image.
- For a 2^m x 2^m image, we have layers 0-m.
- Next pixel at layer k is 2^{m-k} positions apart.
- The “gap” in the next layer (k+1) is half of the gap in layer k.
- For each pixel decoded before layer k, we add three pixels at layer k.

H5A.java, the Encoder

```
public class H5A{
   static int numberOfValues = 256;
   int width, height;  // image dimensions
   short[][][] raw;      // the raw image stored here
```

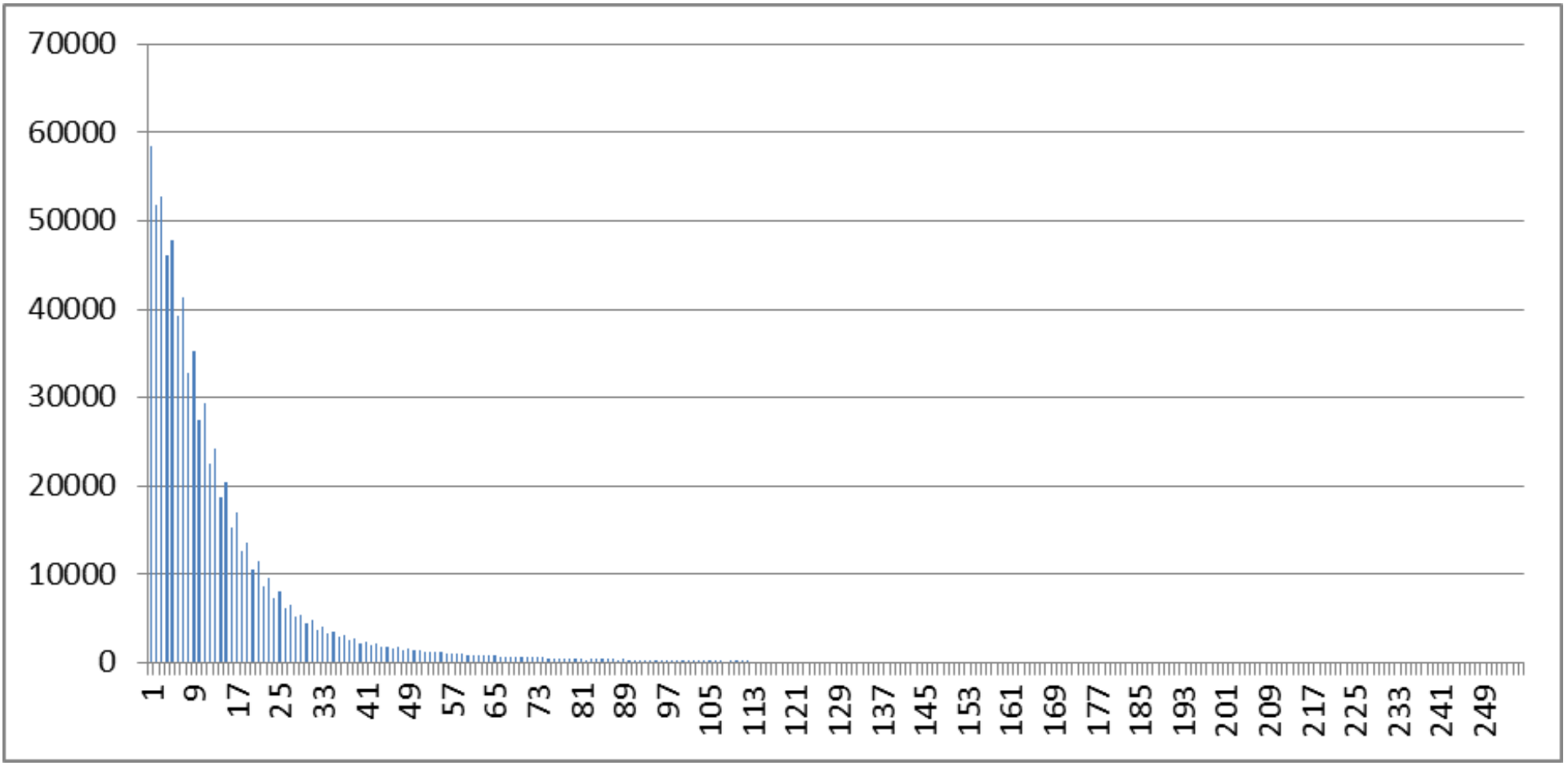
```
      public static void main(String[] args){
         H5A h5 = new H5A();
         h5.readHeader();
         h5.readImage();
         h5.progressive();
       }
```

```
 void progressive(){
   for (int k = 0; k < 3; k++) System.out.write(raw[0][0][k]);
   int size = 1;  int gap = height;
   while (gap > 1){
     int halfGap = gap >> 1;  int rpos = 0; int cpos = 0;
     for (int i = 0; i < size; i++){
        cpos = 0;
        for (int j = 0; j < size; j++){
          for (int k = 0; k < 3; k++){
             System.out.write(mapError(
   raw[rpos][cpos + halfGap][k], raw[rpos][cpos][k]));
             System.out.write(mapError(
   raw[rpos + halfGap][cpos][k], raw[rpos][cpos][k]));
             System.out.write(mapError(
   raw[rpos+ halfGap][cpos + halfGap][k], raw[rpos][cpos][k]));
          }
          cpos += gap;
        }
        rpos += gap;
     }
     gap = halfGap; size <<= 1;
   }
   System.out.flush();
}
```

Same Error Mapping as in H4A

```
// map the prediction error to nonnegatives
  int mapError(int value, int prediction){
     int e = value - prediction;
       //  prediction error
     if (e > 127) e -= 256;
       //  putting error in [-128, 127]
     else if (e < -128) e += 256;
     e = (e >= 0) ? e * 2 + 1 : -e * 2;
       //  into 0 -1 1 -2 2 array
     return e - 1;
  }
```

Lena Progressive Errors Entropy = 5.31



Lena 64 x 64

H5B.java, the Decoder

```
public class H5B{
   static int numberOfValues = 256;
   int width, height;  // image dimensions
   short[][][] raw;      // the raw image stored here
   byte[] data = null;
   int index = 0;



    public static void main(String[] args){
       H5B h5 = new H5B();
       h5.readHeader();
       h5.readData();
       h5.progressivelyDecode();
       h5.outputRestored();
     }
```

```
void progressivelyDecode(){
   raw = new short[height][width][3];
   for (int k = 0; k < 3; k++) raw[0][0][k] = nextDatum();
   int size = 1;  int gap = height;
   while (gap > 1){
     int halfGap = gap >> 1;  int rpos = 0; int cpos = 0;
     for (int i = 0; i < size; i++){
        cpos = 0;
        for (int j = 0; j < size; j++){
          for (int k = 0; k < 3; k++){
             raw[rpos][cpos + halfGap][k] = ?;
             raw[rpos + halfGap][cpos][k] = ?;
             raw[rpos+ halfGap][cpos + halfGap][k] = ?;
          }
          cpos += gap;
        }
        rpos += gap;
     }
     gap = halfGap; size <<= 1;
   }
}
```

```
 void readData(){
   data = new byte[height * width * 3];
   try {
      System.in.read(data);
   } catch (IOException e){
     System.err.println(e.getMessage());
     System.exit(1);
   }
 }
```

```
 short nextDatum(){
// next symbol, maybe from the next codeword
   int x = data[index++];
   if (x < 0) x += 256;
   return (short)x;
 }
```

Homework 5: due 2-2-15

- Complete H5B.java so that it functions as the inverse of H5A.java.
- The same modifications can also make H5C.java work.
- Decode using either H5B or H5C the given test5.bmp that is generated with H5A.
- Submit a document containing your coding and an image of the decoded test5 image.