1. What scheme or schemes did you try? If you came up your own idea, describe it here.

I used differential coding, preprocessing the raw data frame by frame and pixel by pixel. Instead of using the example given in the document, where all pixels in a frame were differentially coded relative to the first pixel in that frame, I encoded a given x, y pixel in a frame relative to the value of that pixel in the last frame. So, instead of encoding each pixel value for every frame, I only encoded the first frame uncompressed. From here, each frame was instead encoded as the difference between the current frame and the past frame. This continues for all frames.

So the decoder took in the first 64x64 pixels in uncompressed then read each symbol as a difference, referring to a dynamically changing “last\_frame” 64x64 array to find what the previous pixel value was. I used a few different distributions (uniform, linear, binomial) to test this scheme.

2. Why do you think your scheme would do a good job predicting pixel values? How does your scheme exploit temporal and/or spatial coherence?

In a video, especially the short and relatively monotonic one that we are encoding, a pixel’s value at any given frame is likely to be similar to the value at the previous frame. This temporal coherence is exploited in my scheme because differential coding between frames should efficiently compress the file if temporal coherence is present. Thus, by taking advantage of the fact that while pixel values reasonably range very widely from 0 to 255, the differences should be heavily centered around 0, with almost no values far from this. This then allows for the encoding of values like -1, 0, 1 etc. to be encoded very efficiently, more so than simply encoding the pixel values themselves. Thus, a binomial distribution whose mean corresponds to 0 would be very effective at predicting pixel values. This is because the binomial distribution would effectively give a high enough frequency to values that we expect to be common for the given source model to encode it more efficiently.

My scheme does not take advantage of special coherence.

3. When applying the English text-based models (static, adaptive, and context-adaptive) to the video data, which scheme performed best? Does the scheme you developed compress better or worse than the English text-based models when applied to video data? If you weren't able to finish and test your own scheme, how do you think your scheme would fare in comparison to the English text-based models?

Between the three English text-based models, they performed as follows: Static 1040 KB, Adaptive 1039 KB, Context-Adaptive 888KB. The best of my schemes uses a binomial distribution and compressed the original 1200KB file to 764KB. So, the scheme I developed compresses much better than the other models.

An important note to make is that I tested my model with multiple distributions before ending on binomial. First I used a uniform distribution for my frequency model, which assumes every difference would appear equally often. This actually increased the file size when encoding the file to 1350KB, because I was not taking advantage of the knowledge that most differences should be near 0 between pixels at the same x, y. Additionally, I used a linear distribution, in which frequency of -255 was 1, increasing to 256 for the frequency of 0, then decreasing again to 1 for 255. This resulted in an encoded file size of 1204KB, which was still greater than the raw input.

These two extra distributions show the importance of the combination of an accurate frequency model along with the differential coding scheme. Because a binomial distribution extremely heavily favors values right around the mean, it perfectly follows the differential scheme.

4. What is one change you could make to your scheme that might improve its results?

One change that I could make would be to calculate the actual frequencies of the differences between the same x, y pixels in subsequent frames, rather than use a binomial distribution. This would require going through every frame transition and also would require the encoding of the frequencies in a file header much like what we did with Huffman encoding, so in the end the improvements over binomial distributions may not be worth it.