COMP 546 HW 3

Kevin McCoy [kmm12]

colab notebook link

I did not collaborate with anyone.

1.0 Lucas-Kanade

1.1

*** see colab notebook ***

1.2

*** see colab notebook ***

1.3

The .gif file can be found here.

Figure 1: gif images

Admittedly, I am disappointed in my result. I believe there must be a bug somewhere in my code, but I have been unable to find it. My guess is that the flow is not being correctly changed when moving between layers of the Gaussian pyramid.

1.4

a Do you notice any inaccuracies in the point tracking? Where and why?

The point tracking does poorly on the guardrails on the left of the video. These objects are clearly not moving, but are fooled by the shadows cast by the cars. This makes sense, as the actual pixel values are changing, causing the points to drift south.

b How does the tracking change when you change the local window size used in Lucas-Kanade?

As the window size increases, the smoother the optical flow is allowed to be. I found that I needed to keep this parameter small, or else all points moved in the same direction as the cars (even stationary points).

2.0 Modified Lucas-Kanade / Conceptual Questions on Segmentation

The matrix **A** is a combination of a shear and scale transformation. This makes the optical flow smoother when the difference between two frames is not a simple translation. For example, this might help with the example of the rotating beer can we saw in class.

2.1

a A higher number of clusters will lead to better segmentations, to a degree. You can imagine at both extremes, having one cluster for the entire image or one cluster for each pixel does not really provide any useful information. Thus, the best amount of pixels is a trade-off between not separating enough items and separating too many items in an image.

b A good number of clusters should be determined on a contextual basis, when one knows about how many groups / items are in or should be in the image. For example, in the beach image, k=20 clusters looks appropriate. It captures the clouds, but doesn't divide them based on their color. It also separates the dry sand from the wet sand. k=50 seems to divide unique items based on whether or not they are in shadow, which doesn't seem too important.

2.2

In this example, it is obvious that spectral clustering is more robust to noise. This is because k-means clustering does not take location into account, which leads to some pixels being placed in the wrong group. Spectral lustering, on the other hand, can include any number of functions into its affinity matrix, including color, distance, and filterbanks.

2.3

By design, Horn-Schunck calculates flow for every individual pixel. However, we can make an assumption that superpixels should have the same flow. This makes sense as superpixels almost always part of one individual object, and thus should all move in the same way. Thus, we can rewrite the objective function to be:

$$\min_{u,v} \sum_{k \in \mathcal{K}} \Big[E_S(k) + \lambda E_d(k) \Big]$$

which sums over each superpixel k, instead of every individual pixel coordinate (i, j). This constrains each superpixel to have identical u and v.

3.0 Image Compression with PCA

3.1

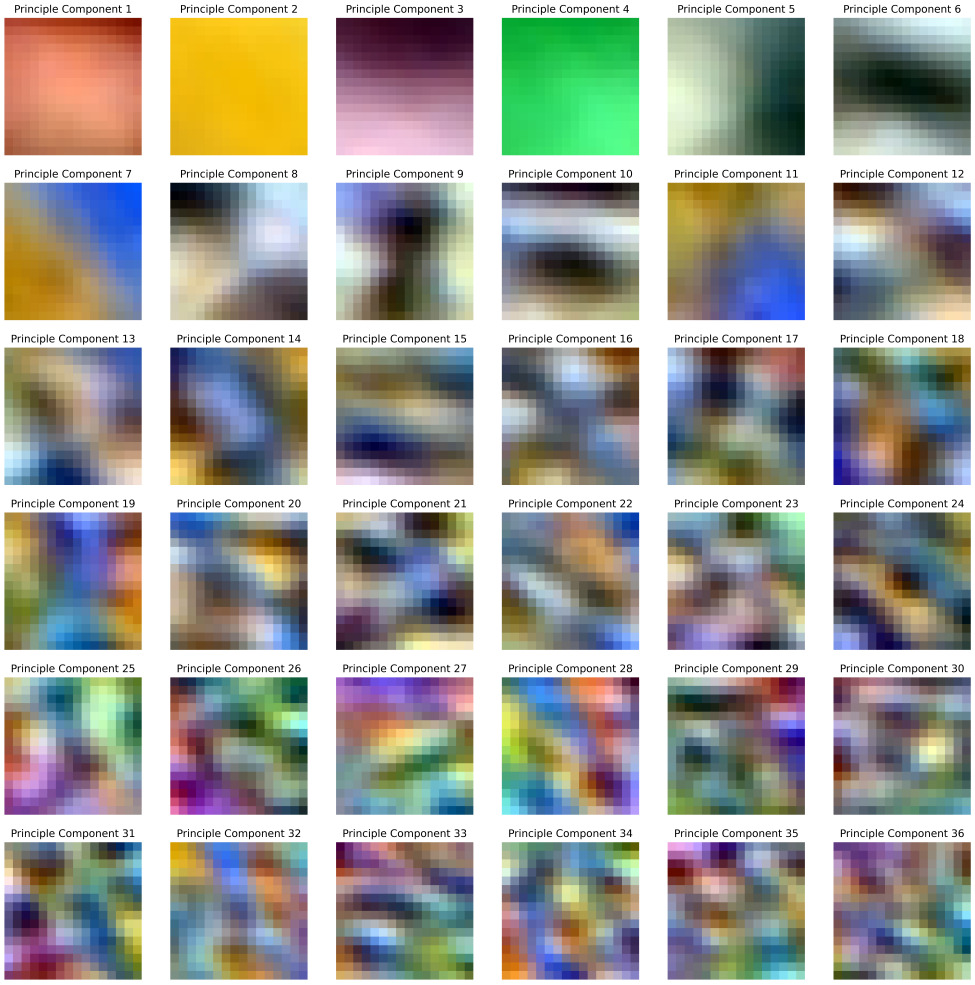

Figure 2: PCA Decomposition of Parrot

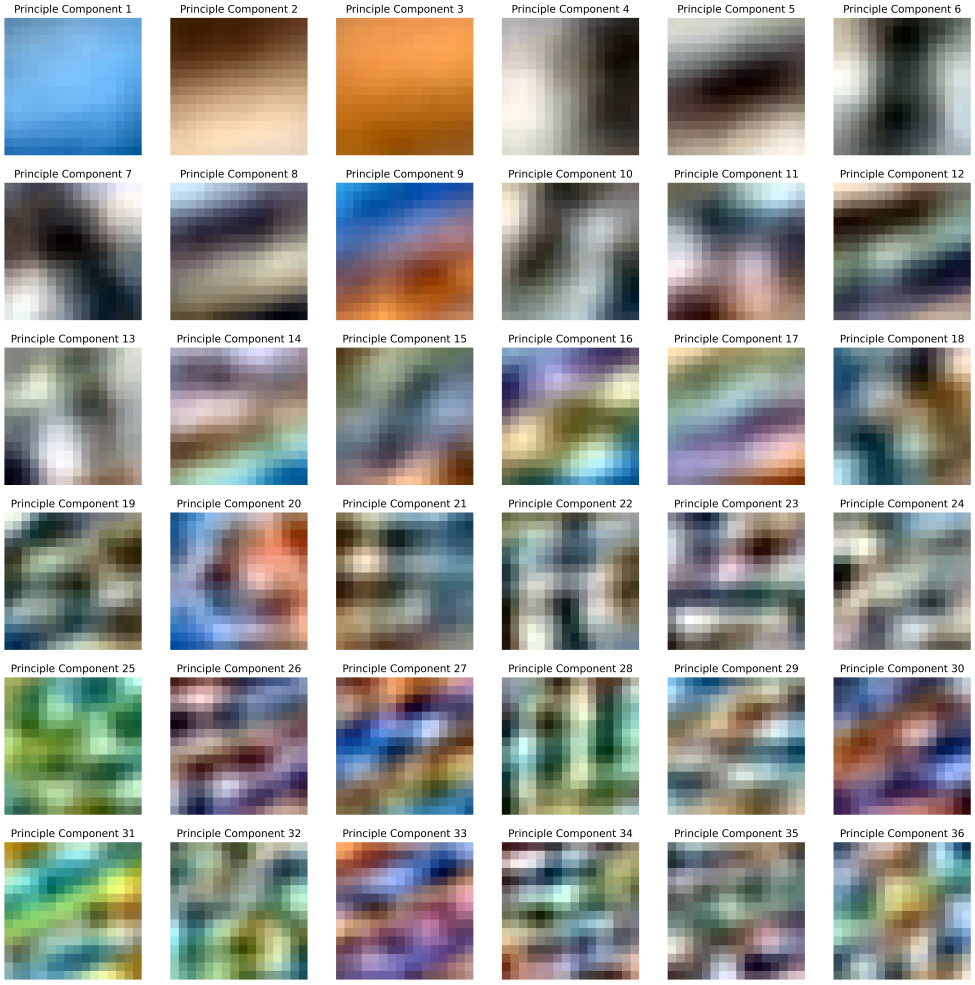

Figure 3: PCA Decomposition of Station

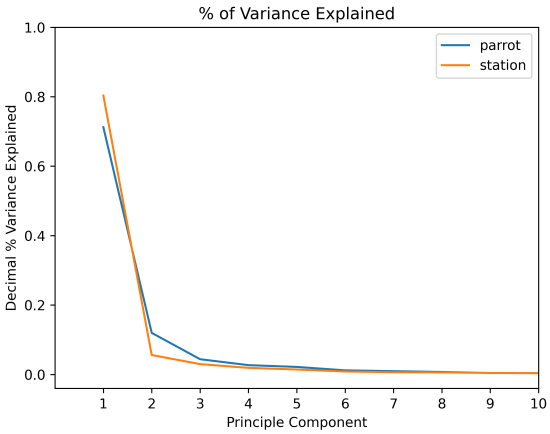

Figure 4: Amount of Explained Variance in each Principle Component

3.2

Figure 5: Parrot Picture Compressed Using Different Number of Principle Components

Figure 6: Station Picture Compressed Using Different Number of Principle Components

1. The train station was easier to compress. We know this because the eigenvectors for the train station decrease faster than those of the parrot, and thus the first few principle components of the station contain relatively more information than those of the parrot. We can also tell this visually, as the parrot becomes blurrier and loses detail quicker (in figure 4 and 5 as we move right to left) as we compress the image further. For example, with 50 principle components the parrot is already losing detail and vibrancy, but the station looks normal. I believe this is because the the parrot has more high frequency details that are hard to contain in a discrete number of eigenvectors. The station image has more low frequency 'details'. The parrot also has a higher dynamic range, and contains much more vibrant colors.
2. Both PCA decompositions start with a uniform vector that likely encodes the mean value of each image. These are then followed by vectors that encode edges, then corners, and so-on. As the eigenvectors go on, they become much more random looking and probably encode higher frequency details.