1 Overview

This section will review some applications using Robust PCA. Currently, most of the applications are related to computer vision. As we will show later, many images involve natural characteristic of low-rank structure, which makes Robust PCA a perfect fit to them. We will also explore some other applications that is theoretically with low-rank and sparse structure and show what we get from them.

2 Robust PCA Applications

2.1 Background modeling from surveillance video

Video is a natural candidate for low-rank modeling, due to the correlation between frames. One of the most basic algorithmic tasks in video surveillance is to estimate a good model for the background variations in a scene. This task is complicated by the presence of foreground objects: in busy scenes, every frame may contain some anomaly. The background model needs to be flexible enough to accommodate changes in the scene, for example due to varying illumination In such situations, it is natural to model the background variations as approximately low rank. Foreground objects, such as cars or pedestrians, generally occupy only a fraction of the image pixels and hence can be treated as sparse errors.

Figure 1: Original video frames

Figure 2: Low-rank components

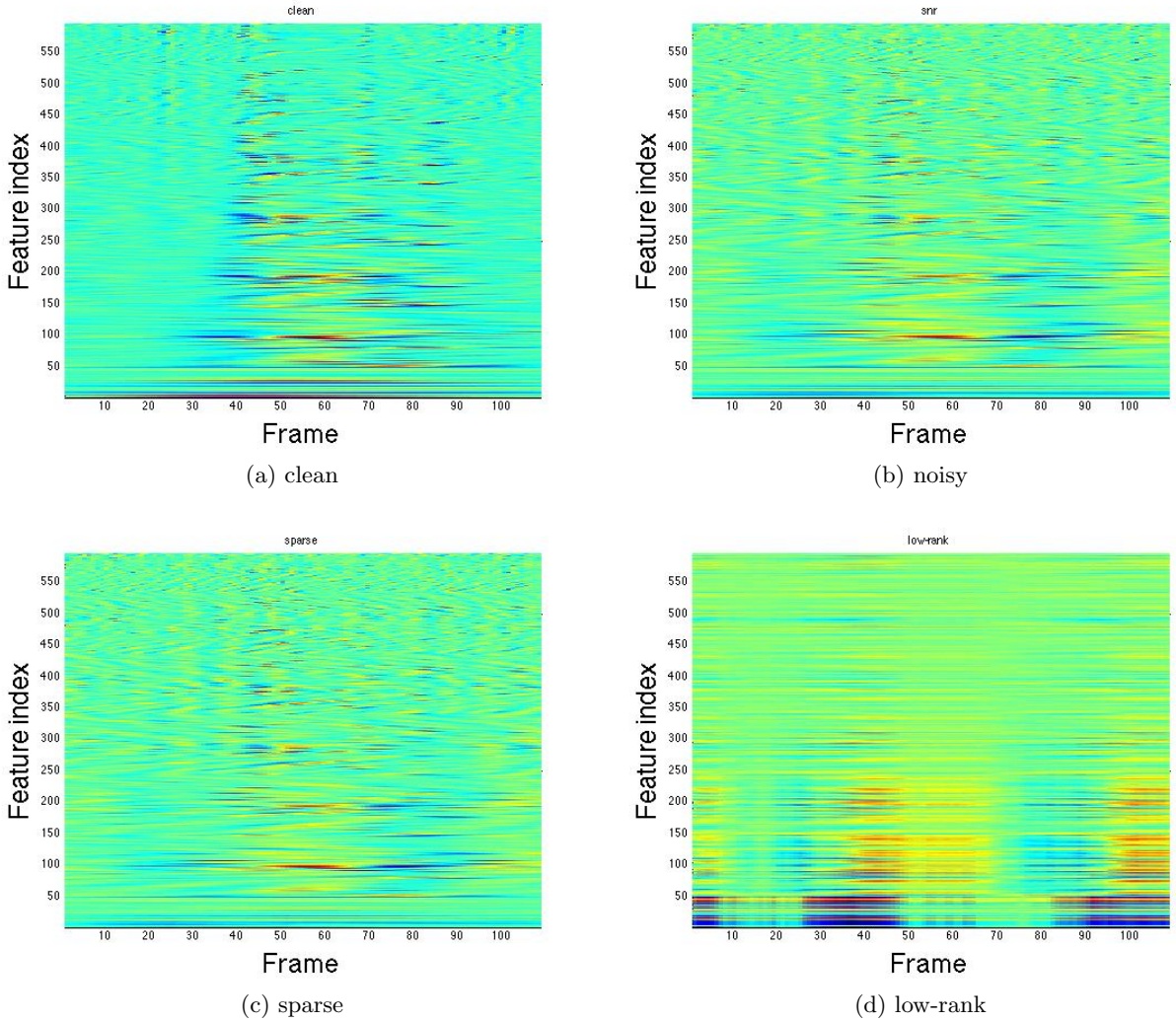
Figure 3: Sparse components

We consider five frames from an original video, as shown in fig. 1, which is a scenario that one man passes by. The resolution of each frame is 176×144. We first separate them into three channels (RGB). For each channel, we stack each frame as a column of our matrix $M \in \mathbb{R}^{25344\times 5}$. We decompose M into low-rank components L and sparse components S by Robust PCA. Then we

combine the three channels again to form images with low-rank component and sparse components respectively. We are using a 2GHz qual core laptop and it takes 0.92s to finish decomposition for three channels. We can find that the L, as shown in fig. 2, correctly recovers the background, while S, as shown in fig. 3, correctly identifies the moving person.

2.2 Using Robust PCA in speech recognition

Intuitively, a consistent sound would be of low-rank structure. If someone is speaking with background noise which is consistent, we would believe that we can separate a clearer speech (the sparse component) from background noise (low-rank component) via Robust PCA framework. If we get a clearer speech signal, the accuracy of speech recognition with current standard technique will increase. According to this thought, we did an experiment and try to see whether Robust PCA works in such case.



(a) clean

(b) noisy

(c) sparse

(d) low-rank

Figure 4: Speech features

Figure 4a shows the features of a clean speech signal, where the x-axis denotes indices of small time frames and the y-axis denote the features vectors at each frame. These features are computed

by a standard method refering to []. Figure 4b shows the features domain of the same speech signal subjecting to noise with $SNR = 0$ in this case. We apply Robust PCA to decouple these features into low-rank component, as shown in fig. 4d, and sparse component, as shown in fig. 4c. Technically, we would believe that sparse component is corresponding to our speech signal whereas low-rank component is corresponding to noise. Then we use the sparse component as new features and train a classifier via method in []. Unfortunately, comparing to the original method without Robust PCA, we only got improvement in data set with subway noise, which we consider as the most consistent noise among the data set. In this data set, Robust PCA improve the accuracy rate with 3% comparing to original method. For other noise, such as street noise and car noise, we both got decrease about 7% in accuracy. One possible reason is that in the real world, most of the noise is not low-rank and they are not consistent. However, speech signal could be low-rank sometimes because there are only a limited vowels and consonants in a language, which makes Robust PCA not so suitable for de-noising.

2.3 Filtering of Brain-Machine Interface neural spike data

We have used Robust PCA on some neural spike data that was collected from a monkey's brain. The Brain-Machine Interface project investigates how neural spike data can be used to interface with machines, with the ultimate goal of establishing some sort of bi-directional link (that is, to be able to control a machine through brain activity and, conversely, to "feel" feedback from measurements obtained by the machine).

The particular problem we have looked at in this project was to use Robust PCA to analyze the neural spike data obtained from the motor cortex of a monkey, who was moving his arms to control a computer cursor in different directions. The assumption was that the measurements would consist of a somewhat "low-rank" component associated to the movement required by the particular task, and some superimposed sparse noise associated to neuron firings in the background. The data available was time-series neural spike data from a particular experiment, where the monkey had to move the cursor in 8 different directions, according to 45 degree increments. Each of the task was repeated a certain number of times. A total of 127 parallel measurement channels were available, each corresponding to a neuron (or a small number of neurons).

As a pre-processing step, we normalized the time over the different trials for each component. The reasoning here was that the same task will correspond to the same neural firing pattern, independent of whether it was performed faster or slower[1]. The next step was to group each of the trials for each of the tasks into a number of time bins, i.e. if the number of bins is N then the i-th bin associated to neuron k counts the number of firings of neuron k within the interval $[(i-1)/N, i/N]$ (recall that trial time is normalized to 1).

Figure 5 shows the raw data after binning with 3 bins. here each of the 25 columns in each task represent the binned data from a single trial, where the 3 bins are stacked vertically. That is, within the large column associated to the 0 deg task, the first three entries of the first column correspond to the three time bins of the first neuron. The data is for 127 neurons, hence the overall matrix has $3 \cdot 127 = 381$ rows. From the raw data in Figure 5 we can already see some interesting features in the data, namely that each of the tasks seems to have its characteristic pattern.

[1] very little is known at this point about the structure of the dataset, so it was up to us to make reasonable assumptions about the data

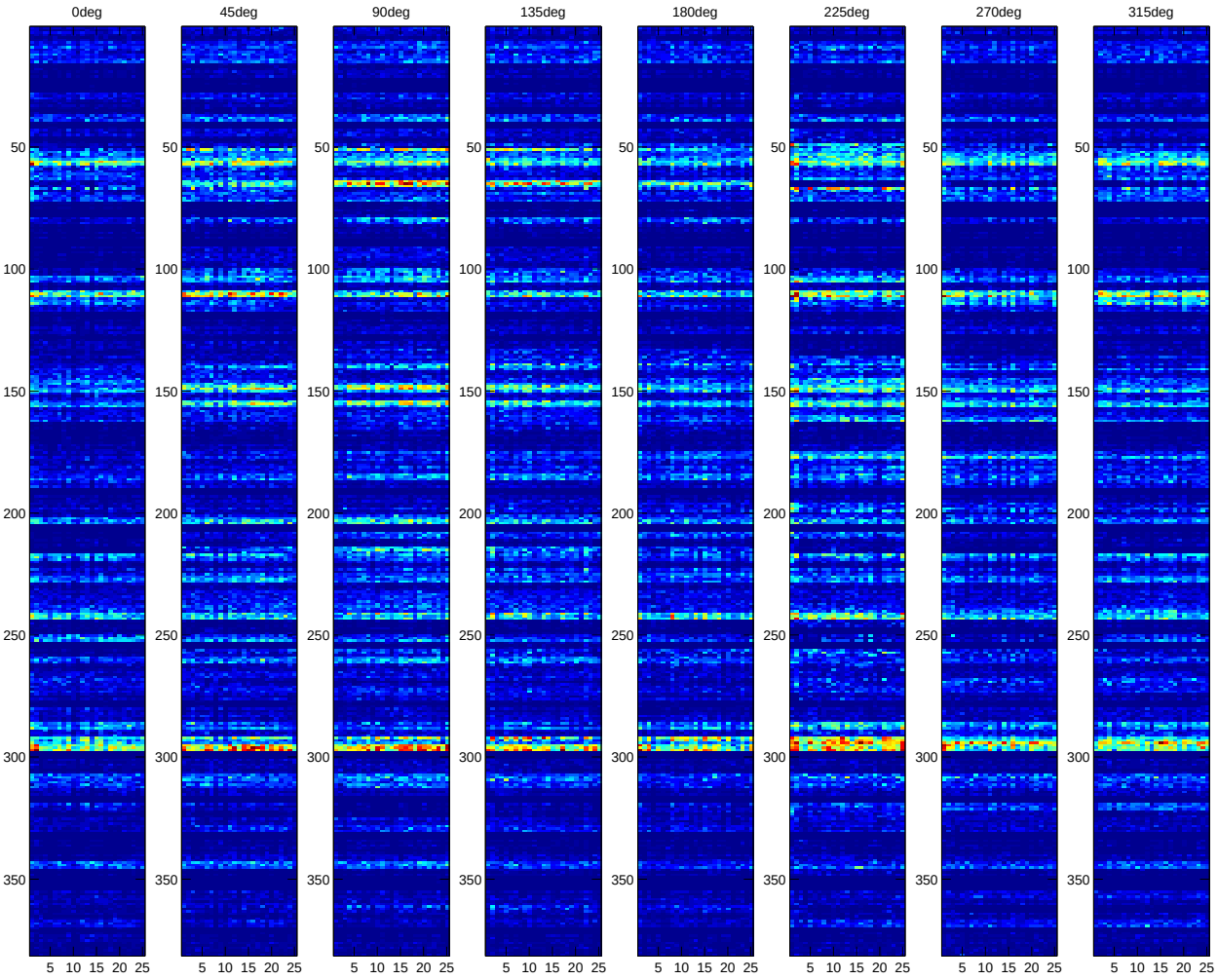


Figure 5: Raw (normalized and binned) neural spike data, $n_{bin} = 3$

For each of the tasks we apply Robust PCA to the data matrix to extract low-rank and sparse components. Figure 6 shows the low rank component extracted from the raw data of Figure 5. We notice that, as expected, the result is a "filtered" version of the raw data, in which the principal characteristics are retained while a sparse component has been removed. We think that the low-rank component in each of the tasks corresponds to the neuron firing pattern that can be directly related to the task, while the sparse component may be mis-detections, firings due to other movements, or simply sparse "background" firing that is always present.

For the 90 deg task Figure 7 shows a direct comparison of the raw data and the low-rank and sparse components obtained via Robust PCA. While for this comparably small data set the basic characteristics can be extracted visually just by looking at the matrix, this in general is not possible for high-dimensional data. One issue that we see with the available dataset is that the low rank component itself is quite sparse (since we are considering time-series data), hence some technical assumptions such as the incoherence condition will not necessarily hold. However, since not even neuroscientists have a clear understanding of what the data characteristics are and what the different patterns mean, we will not attempt to resolve this issue here. After the end of the semester we plan to learn more about the Brain-Machine interface project and, together with domain experts,

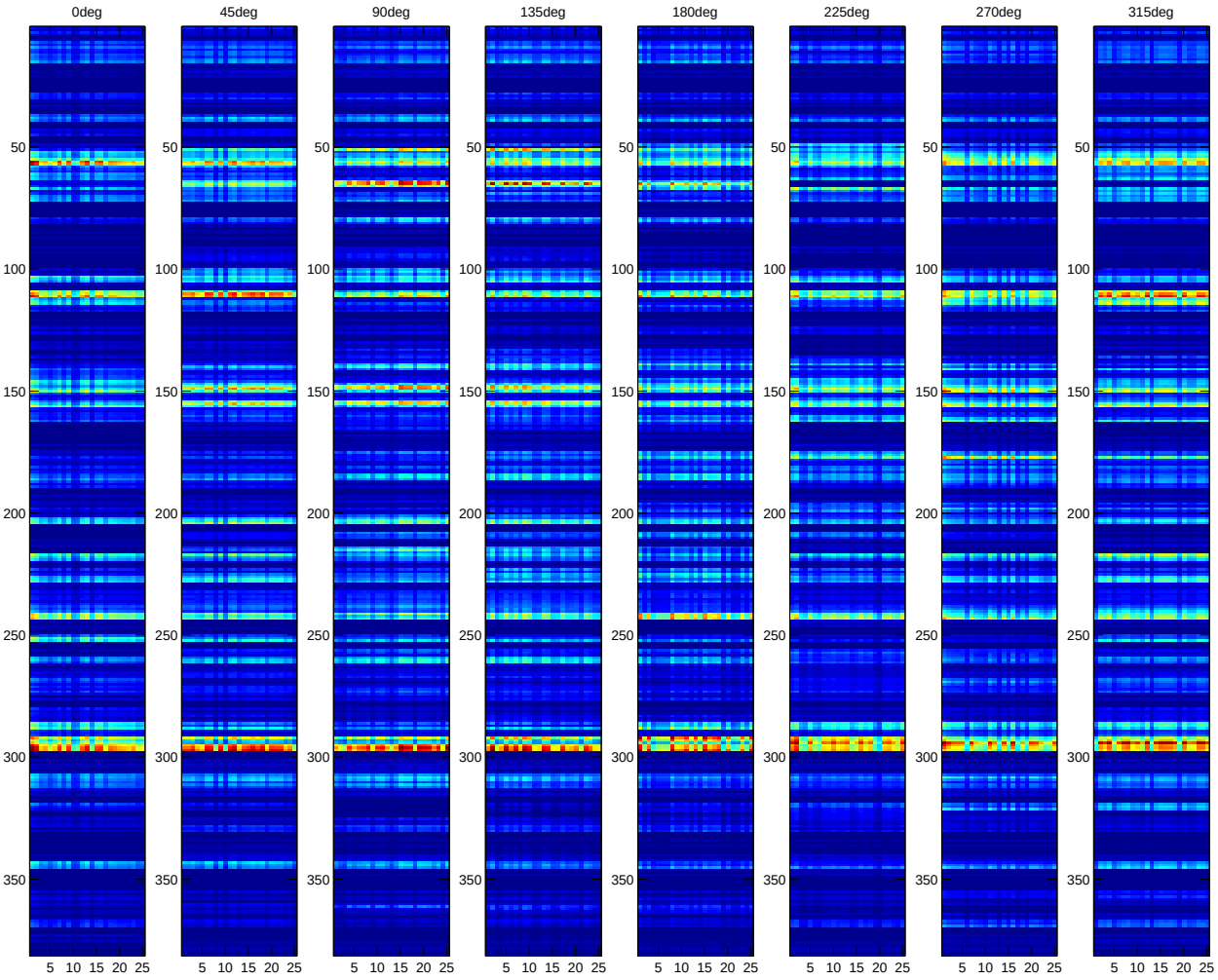


Figure 6: Processed neural spike data, $n_{bin} = 3$

to identify possible problems where Robust PCA could be used.

Based on the difference in neurons firing patterns between different tasks one can use machine learning techniques to predict from neural measurements which task the monkey is performing. Here Robust PCA could be used as a pre-processing step to filter out the sparse noise component. However, due to our lack of domain knowledge it is at this point unclear whether the Robust PCA framework has advantages over other and possibly simpler methods, for example a simple low-pass filters along the rows of the data matrix. Our main reason for presenting this application example here was to illustrate that Robust PCA can indeed be used as a tool to analyze interesting real data of reasonable size.

3 Discussion

Robust PCA framework is a powerful tool to separate low-rank components and sparse components if we are given a combination of these two structures. However, generally most of the real world data is not directly a combination of these two structures unless the data is subjected to some

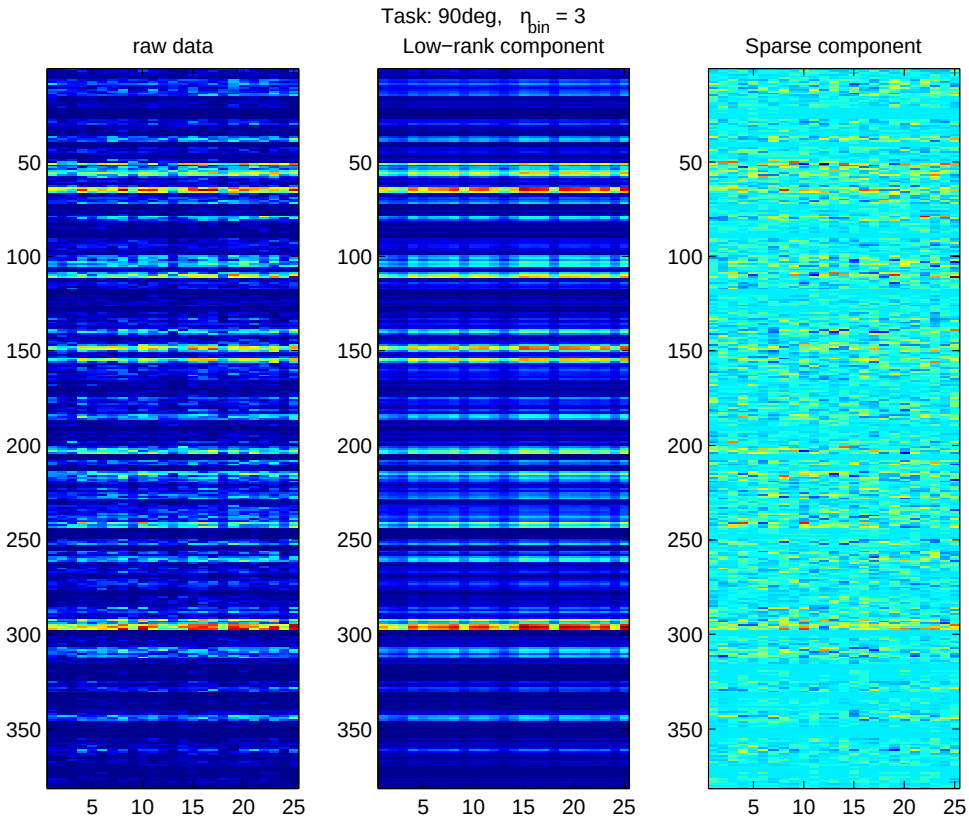


Figure 7: Raw data, low-rank component L and sparse component L for 90 deg task, $n_{bin} = 3$

kinds of transformation. One example is the data set of human faces with different expressions and illuminations. In this case, some researches have proposed some methods, RASL [1] and TILT [2], based on the original convex optimization problem in Robust PCA. They introduce transformation parameters to the original framework and by linearizing with respect to the transformation parameters, we got a new but not difficult convex problem. As long as the data is within a certain degree of transformation, the algorithms can still recover the low-rank structure and sparse structure.

We can also observe that is image data, the sparse noise is always presented as blocks. For example, the person in fig. 3 appears in connected pixels, so the noise is not distributed randomly in pixels. A possible direction that can improve the separation in image data is that we can introduce a penalty term in the original optimization problem:

$$\begin{aligned} p^* = \min_{L,S} \; & ||L||_* + \lambda |||S||_1 + \beta \sum_{i,j \in \mathcal{N}(i)} (S_i - S_j)^2 \\ \text{s.t.} \quad & M = L + S \end{aligned} \tag{1}$$

where $\mathcal{N}(i)$ means all the entries next to i in matrix S. This problem is still convex so we can solve it via existing algorithm for convex optimization problems.

References

[1] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma. Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 763 –770, june 2010.

[2] Z. Zhang, X. Liang, A. Ganesh, and Y. Ma. Tilt: Transform invariant low-rank textures, 2011.