Reducing signal contamination in video files.

In this project, we investigated brain activity over the brain region responsible for sound processing in a model with a specific gene removed. The purpose of this scientific research was to gain insight into the role of the gene in brain functions. In part 3 of this project, we used a combination of principal components analysis (PCA) and independent components analysis (ICA) to identify and remove contamination in the brain signals in our video files. This project forms a part of my PhD thesis, in which the complete content can be found in the University of Auckland Doctoral Thesis repository.

Introduction:

In this scientific study, we measured brain activity in the form of changes in fluorescence intensity over time. When a brain area is bright, this indicates high brain activity, and areas that are dimmer indicate low or a break in brain activity. These signals are recorded in a video file, and then we extract data from the video files for analysis (data extraction described in part 1).

Ideally, all the brightness we are recording should be solely from brain cells (the source of brain activity). However, in reality, brain cells and blood proteins (haemoglobin) emit fluorescence. This meant the signals we recorded are a combination of signals from brain cells and blood proteins. Therefore, we need to find a method to identify and remove signals from blood proteins so that we are analysing signals only from brain cells.

To do this, I used a combination of principal components analysis (PCA) and independent components analysis (ICA) to identify and remove contamination in the brain signal. This combination of PCA and ICA was described in previous studies[1,2], which I have adopted and used for my own data. The use of PCA was to reduce dimensionality, and ICA was used to separate the signals to where they were sourced from (in this case, the signal should be coming from specific regions of the brain[3]).

For simplicity, rather than using biological images, I will use scenery images. The purpose is for a better understanding of what is happening to the images, and it is easier when we know what we are seeing. The same concept is applied to my biological images in my PhD.

---

[1] Makino H., et al. (2017). Transformation of cortex-wide emergent properties during motor learning. *Neuron, 94(4). 880-890.e8*. https://doi.org/10.1016/j.neuron.2017.04.015

[2] Ren, C., Komiyama, T. (2021). Wide-field calcium imaging of cortex-wide activity in awake, head-fixed mice. *STAR protocols, 2(4), 100973*. https://doi.org/10.1016/j.xpro.2021.100973

[3] Each region in the brain is responsible for a specific process, e.g. sound, vision, memory and emotion.
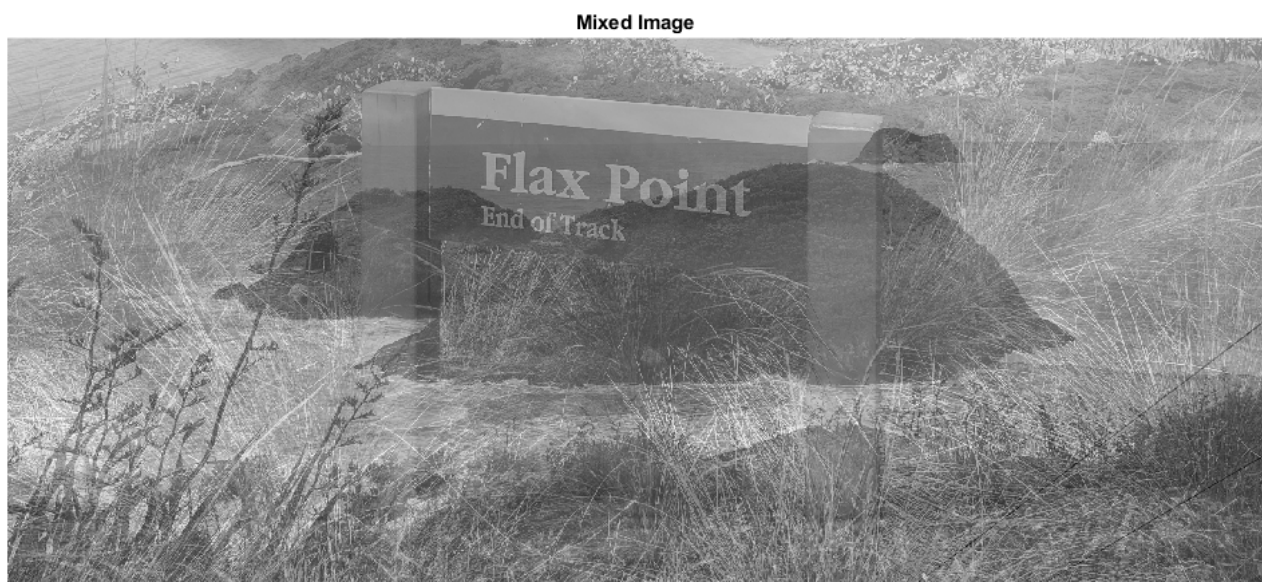
All processing and analyses were done on MATLAB. In MATLAB, the core Python libraries (*numpy*, *pandas* and *matplotlib*) are built into the environment, and therefore we do not need to import these libraries. However, we will be needing specialised functions from the *Statistics and Machine Learning* and *Image Processing* toolboxes.

Principal Components Analysis (PCA)

Load the image into MATLAB. Mimicking a video file, which is a series of images (frames) stacked together, the mixed image we will be working with was created by a stack of three images. In MATLAB, the function 'pca' will centre the data by default.

[1]

```
>> % Load image into MATLAB
>> I = tiffreadVolume('Mixed_Image.tif');
>>
>> mixed_view = mean(I,3);
>> figure
>> set(gcf,'color','white')
>> imshow(mixed_view,[]);
>> title('Mixed Image')
```


The dimension of this image is $x*y*z$, where x is the number of pixels in the x-direction and y is the number of pixels in the y-direction (which forms an image for one frame), and z is the number of frames in this file. For PCA, we have to transform the 3D data into a 2D matrix where pixels are the observations (row), and frames are the features (column).

[2]

```
>> % Transform the 3D image data into a 2D matrix
>> X = reshape(I, size(I,1)*size(I,2),size(I,3));
>>
>> % Get the mean used to centre data
>> Xmean = mean(X,2);
>>
>> %% Principal components analysis
>> [coeff, score, ~,~,explained] = pca(X);
>> % coeff = loading coefficient for each feature; row = features; col = PC,
>> % score = the score of the variable; row= pixel; col = PC
>> % explained = the percentage of variance explained; row = PC
```

In this example, the size of the image file is small, with 3 frames, meaning it is obvious to determine the number of principal components (PC) needed for independent components analysis. In a video file, the size would be much bigger, with 1000+ frames. With this size, we need the number of PCs that will explain 90% of the variance because we want to retain as much information as possible while reducing dimensions.

We will not be visualising each PC because it is irrelevant for our task. Each PC is ranked from most to least variance explained. In images, the greatest variance would come from the intensity/brightness in the image, while small details, such as texture, shape and edge, give smaller variance. In an image, all are important, but in terms of PC, they are ranked. Therefore, PCA was used solely for dimensionality reduction.

Independent Components Analysis (ICA)

We will put PCs into an ICA algorithm to separate the features into statistically independent (not correlated) sources. What we aim to do is to separate the features in the mixed image into their independent sources. We are using a statistical method to see where each part of the mixed image came from[4]. The ICA algorithm used is the Joint Approximate Diagonalisation of Eigenmatrices (JADE)[5] which uses real-valued signals. Each independent component (IC) is not ranked, but are equally important and represents the components that makes up the data/mixed image.

---

[4] To use ICA, our data must meet the following assumptions. A. The data is a mixture of independent sources. B. The sources are independent, in other words, each source does not affect each other, and C. The data cannot be Gaussian.

[5] Cardoso, J. F. (1999). High-order contrasts for independent component analysis. *Neural Computation*, *11*(1), 157–192. https://doi.org/10.1162/089976699300016863

[3]

```
>> %% Independent components analysis
>> % The output B is the separating matrix.
>> % The output W is the whitening matrix.
>> % The score data was transposed because under the JADER documentation, the
>> % input data should be organised as measures (row) and samples (col).
>>
>> [B W] = jader(scorePCA');
>>
>> % Unmix the PC score using the separating matrix
>> scoreICA = (B*scorePCA')';
>>
>> % Putting the IC score back into image shape
>> IC = reshape(scoreICA,size(I,1),size(I,2),[]);
>>
>> figure
>> set(gcf, 'color','white');
>> subplot(3,1,1);
>> imshow(IC(:,:,1),[])
>> title ('IC 1')
>> subplot(3,1,2);
>> imshow(IC(:,:,2),[])
>> title ('IC 2')
>> subplot(3,1,3);
>> imshow(IC(:,:,3),[])
>> title ('IC 3')
```

The original MATLAB script for JADE only returns *B*, the separating matrix. As we need the whitening matrix, W, we have to edit the script to include the line below right before the `return` command at the end of the script.

```
>> W = diag(1./scales)* U(1:n,k(rangeW))';
```
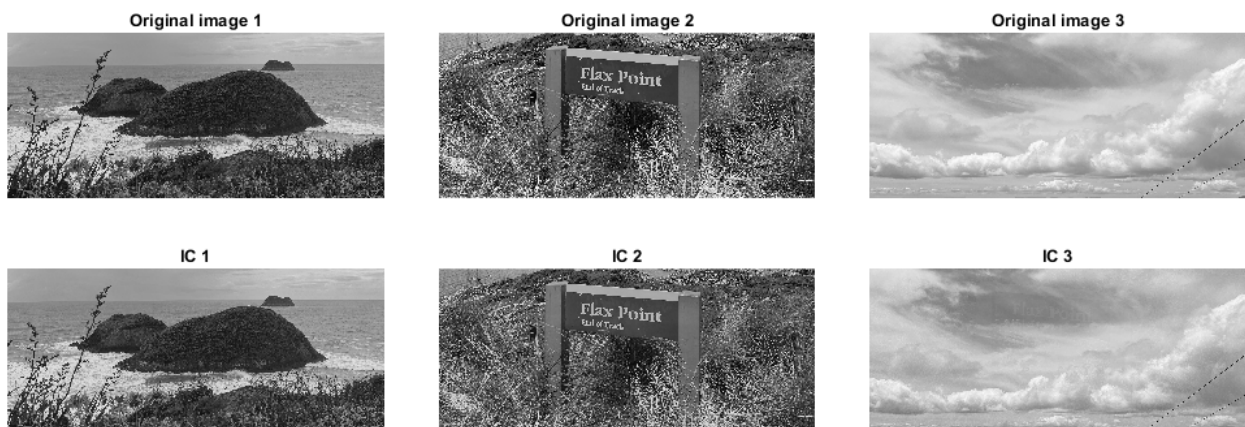
IC 1

IC 2

IC 3

We can see that the ICA algorithm did a good job in separating the pixels from the mixed image into three separate and clearly distinct images. Usually, we are blinded from the original/source images that were used to create the mixed image. In this scenario, however, we do. Let's compare the original and the IC images.

[4]

```
>> % Comparing IC and the original images
>> figure
>> set(gcf, 'color','w')
>> tiledlayout(2,3,'TileSpacing','compact')
>> nexttile
>> imshow(OG1,[]);
>> title ('Original image 1')
>> nexttile
>> imshow(OG2,[]);
>> title ('Original image 2')
>> nexttile
>> imshow(OG3,[]);
>> title ('Original image 3')
>> nexttile
>> imshow(IC1,[]);
>> title ('IC 1')
>> nexttile
>> imshow(IC2,[]);
>> title ('IC 2')
>> nexttile
>> imshow(IC3,[]);
>> title ('IC 3')
```


Original image 1   Original image 2   Original image 3

IC 1   IC 2   IC 3

We can see that each independent component did a good job in separating the pixels into its independent sources. However, upon closer inspection, we can see that it is not a complete recreation of the original image (clear example is the IC3 where we can see ghosting of IC2). This is expected because we are using a statistical method (or an algorithm) to find different pixels in the mixed image. The algorithm is to look for independent components and not to recreate the image.

Reconstructing the images with specific components removed

If we were to identify what were the components that made up the mixed image, the ICs alone were good enough. However, in my PhD, I had to identify components of brain activity signals and components where the signals were from blood in the video files. Then the signals from blood were removed from the video file so that the reconstructed video used for analysis will be signals coming from brain cells.

Again, for purpose of simplicity, I will proceed with the reconstruction of the image this time without IC 2.

[5]

```
>> %% Reconstructing images.
>> % To remove contamination from the original image, by
>> % remove the unwanted IC so that all the wanted IC are left.
>> % We will then reconstruct the remaining ICs to create a whole image, but
>> % with some IC removed.
>>
>> % get the mixing matrix
>> A1 = pinv(B)';
>>
>> % Retain only the wanted IC by removing the unwanted IC in both the
>> % mixing matrix and the IC score
>>
>> % Clean mixing matrix; keep wanted IC
>> A = A1([1 3],:); % row: IC
>>
>> % clean IC score
>> S = scoreICA(:,[1 3]);
>>
>> % Convert IC back into the PC space
>> PC_clean = (S*A);
>>
>> % De-whiten the data
>> PC_deW = (pinv(W)*PC_clean')';
>>
>> % Convert back to the original space
>> Xclean = (PC_deW)+Xmean;
>>
>> % Reconstructing into an image
>> rec_mixed = reshape(Xclean,size(I,1),size(I,2),[]);
>> reconstruct = mean(rec_mixed,3);
>>
>> figure
>> set(gcf,'color','w')
>> imshow(reconstruct,[])
>> title('Reconstructed image')
```


Reconstructed image

We have successfully removed pixels from IC2 from the mixed image.