# Hand Rotation Detection in Video

Magson Gao

## Abstract

As smart home appliances increase in popularity, gesture based control systems become a more viable solution to the human-computer interaction problem. However, many devices of today rely on expensive dedicated hardware and digital logic in order to accurately detect hand motion. In this work a highly efficient implementation of hand rotation detection is proposed which can provide adequate latency on traditional computer hardware.

## Introduction

The system developed in this work uses hand motions to pause, resume and adjust music volume. It does so by reducing the dimensionality of USB camera data using a sobel filter and 2D-Discrete Fourier Transform (2D-DFT) so that it can be processed by a bi-directional Long Short-Term Memory (bi-LSTM) neural network.

## Method*

### Histogram-Equalization

This work aims to function in a diverse range of lighting environments. As the camera used does not have any auto-white balancing, it was necessary to apply histogram equalization to the black and white image. This allows edges to be detected even in over or under-exposed environments. The effect of histogram equalization is shown in Figure 1.

### Edge-Detection

The rotation of objects in an image can be detected using the edge information only. This allows for the amount of data processed by the rest of the system to be reduced significantly. To reduce the amount of data present in the image the edge detection algorithm converts the 24 bit RGB image (8 bits for R, G and B) to 1 bit per pixel. The edge detection algorithm chose was the Sobel kernel due to its computational simplicity and robustness to noise [1]. The 3-by-3 Sobel filter kernel is given below:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

The 3-by-3 Sobel filter kernel can be decomposed into a multiplication of length 3 row and column vectors, thus requiring a single multiply by 2 per output pixel. Multiplies by 2 can be considered as a logical shift operation and multiplies by -1 can be considered as a bit inversion operation and an addition, thus the X and Y Sobel kernels can be computed in very few clock cycles

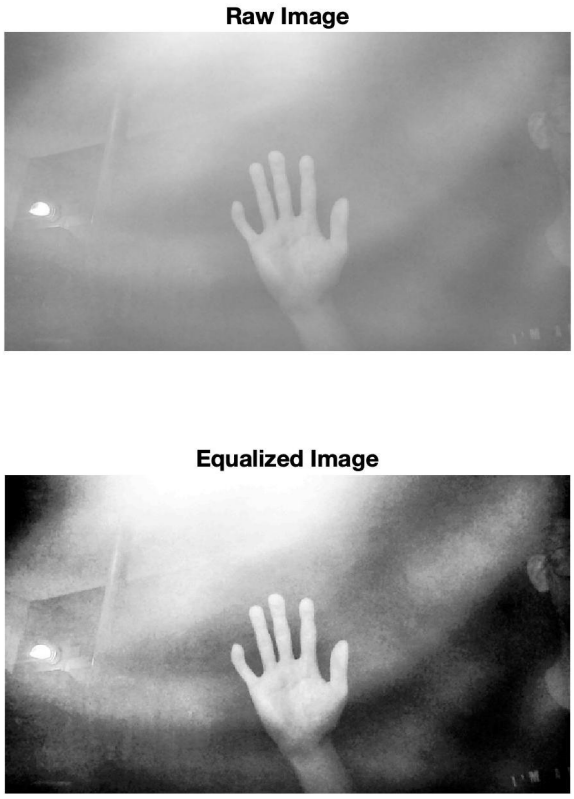


**Figure 1: Top:** Raw 720p image from camera. Banding due to light source causes the edges of the hand to be fainter. **Bottom:** Histogram equalized 720p image.

in software. The Sobel filter was applied to the blue color channel exclusively. This was chosen as most skin colors have lower blue components compared with red and green channels. As most light from the environment (such as the screen from which the camera was attached to) has a high blue component, the absence of blue component in skin produces distinct boundaries upon Sobel filtering. The result of Sobel filtering a 1280 by 720 pixel image is shown in Figure 2.

The histogram equalization operation produces banding artefacts and salt-and-pepper noise. Whilst the latter can be removed by median filter, the former cannot, thus a 5-by-5 Gaussian blur filter chosen to remove the noise before the Sobel filtering. Unlike median filtering, the Gaussian filter preserves edge shapes. The result is shown in Figure 3.

### 2D-Discrete Fourier Transform

Edges in the image produce sharp peaks in the magnitude spectrum of the Fourier Transform domain. This is utilized when detecting hand motion. As each digit of the hand results in lines in the spatial-domain,

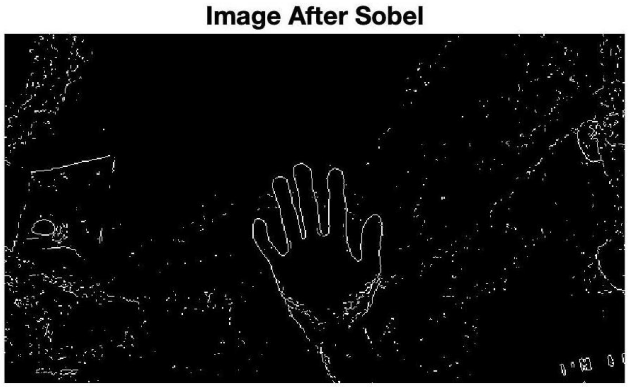


**Figure 2:** Image from Figure 1 after Sobel filtering.

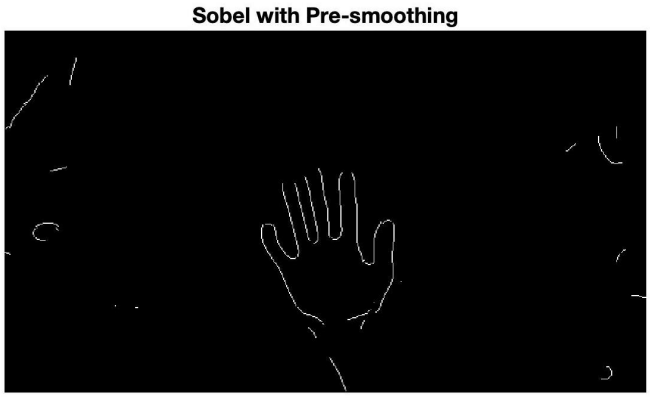


**Figure 3:** Sobel filtered image after Gaussian blur.

the resulting 2D-DFT magnitude spectrum possessing peaks in the perpendicular direction. An advantage of using the 2D-DFT magnitude spectrum is that the magnitude is shift invariant, thus the location of the hand in the image has no effect on the intensity of the resulting magnitude peaks. The spectrum for the Sobel filtered image is shown in Figure 4. The 2D-DFT includes lower value peaks associated with average intensities of spatial frequencies of the image. As the method described in this work relies on the visibility of the peaks, bins of intensity below the 3rd quartile are set to 0. The resulting spectrum is shown in Figure 5. The 2D-DFT can be implemented efficiently using the 1D-FFT requiring $\mathcal{O}(N \log N)$ operations [2].

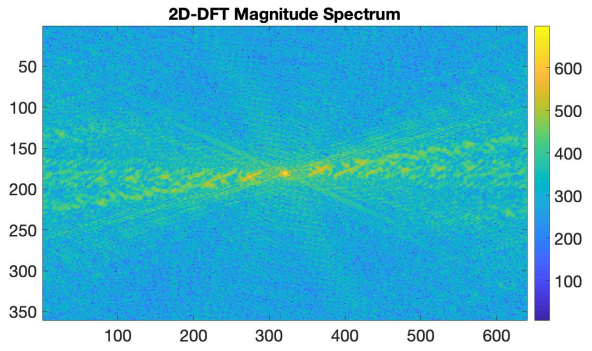


**Figure 4:** 2D-DFT magnitude spectrum. Magnitude plotted is normalized for visibility.

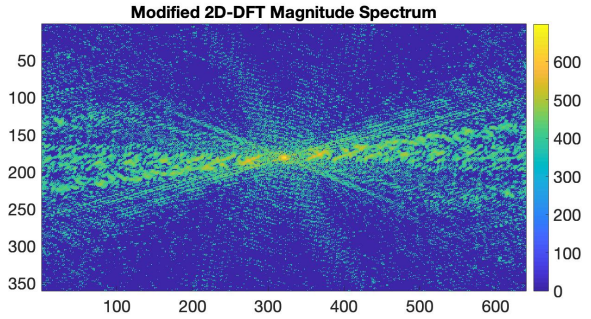


**Figure 5:** Modified 2D-DFT magnitude spectrum. Magnitude plotted is normalized for visibility.

### Remapping from 2D to 1D

To reduce the dimensionality of the data the 2D-DFT spectrum is mapped so that intensity peaks are mapped to linear peaks at an X coordinate corresponding to the angle through the origin. The first part of this process is to apply a polar to cartesian transformation on the 2D-DFT. It was determined experimentally that a width of 256 pixels provided a sufficient angular resolution to detect hand rotation and a vertical resolution of 64 pixels was able to encode sufficient information about the pattern of peaks at higher frequency. The mapped 2D-DFT is shown in Figure 6. This step requires a linear interpolation for each output pixel, thus requires $\mathcal{O}(MN)$ computations.

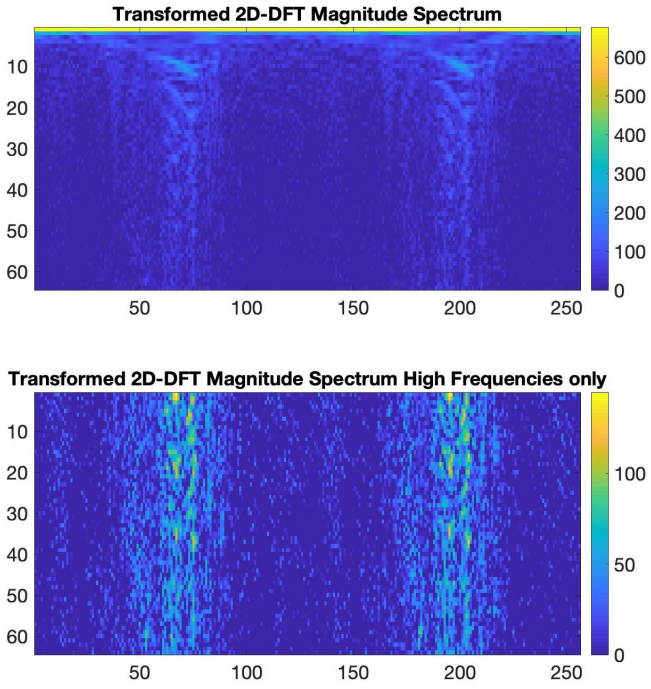


**Figure 6:** Transformed 2D-DFT. Each X coordinate represents the intensity spectrum of the 2D-DFT through the origin in the angular region of width $\frac{2\pi}{256}$. The low frequency information did not contain any useful information thus only the bottom half of the transformed image corresponding to the higher frequencies were used.

To find the peaks corresponding to angles perpendicular to the digits in the Sobel filtereed image, the transformed intensity map was summed in the Y direction. Due to noise which is let through the sobel filter the resulting summation is smoothed before further processing. Both plots are shown in Figure 7. For computational simplicity a 32 tap moving average filter was chosen as the smoothing function. The 2 peaks in the smoothed output corresponds to the presence of lines at angles perpendicular to the digits in the 2D-DFT magnitude spectrum.

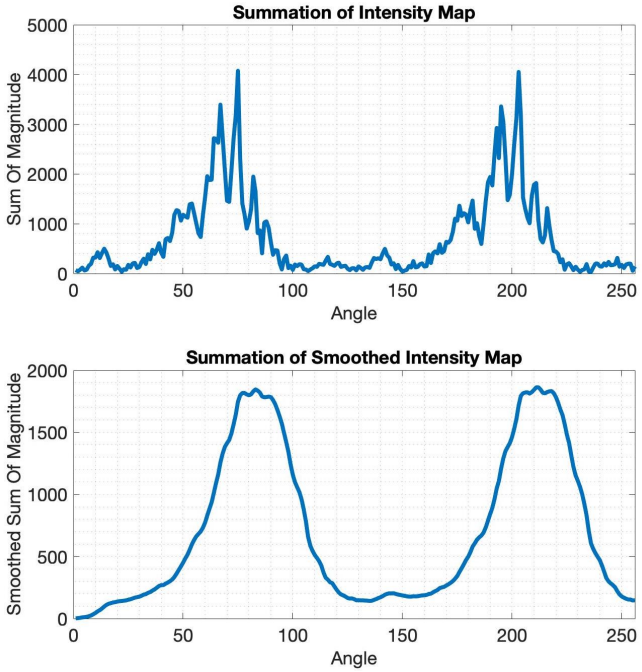


**Figure 7:** Summation of the transformed 2D-DFT.

The final step to convert this to a 1D data structure was to integrate blocks of the smoothed summation. The values in the 1D array for the processed frame is shown in Figure 8.

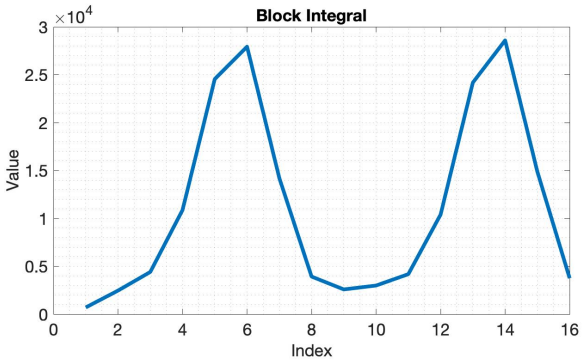


**Figure 8:** 1D array values for the processed raw image.

### Removing Stationary Objects

In many environments lines are visible in the Sobel filtered image due to objects which are not the hand in motion. An assumption can be made in most use cases that the camera and scene is approximately stationary with the exception being the motion of the hand. Thus a buffer of size 64 of the mean array values for previous frames is held to determine whether or not the peaks from the frame being processed currently contains new peaks which is caused by hand motion. As the array is 1D, this significantly reduces the memory requirement compared to storing the images of previous frames.

### Deep Learning

A classification accuracy of 70% was obtained using empirically defined threshold parameters based on the motion of the array values alone. A neural network was used for greater classification accuracy.

### Generating Training Data

In order to produce training data 100 short videos of approximately 10 seconds each were captured of a subject rotating their hand left and right. The subject was given a set of randomly generated sequences of hand motions to complete. There was a short 2 second delay between changes in motion instructions to allow the subject to change motion. To prevent possible overfitting, the videos were taken with a variety of backdrops and in different lighting conditions.

The number of videos was doubled by flipping each video frame horizontally and inverting the rotation labels from left to right and vice versa. The resulting set of videos was doubled again by time reversing each video and inverting the rotation labels from left to right and vice versa. This ensured that the training set had even distribution of left and right labels. To prevent data leakage rotated and time reversed videos were placed in the same set (training or validation) as the original video.

### Deep Learning Architecture

Before feeding the data into the a neural network, spacial and temporal differences were calculated to reduce the training time. This calculation requires one subtract the array values for the current frame from the circular shifted values of the array in the previous frame. The input to the neural network is thus dimension 48.

The percentage accuracy and cross entropy loss for different training schemes over 200 epochs and a mini-batch size of 128 is shown in Table 1. It was determined that 10 hidden units was sufficient to model the data, resulting in a high training accuracy without over fitting.

The network architecture is shown in Figure 10. A bi-LSTM was chosen to model the time dependent data. The bi-directional nature was chosen due to its greater modelling flexibility and faster convergence [3]. Dropout layers and a L2 regularization coefficient of up to $7.5 \times 10^{-4}$ were experimented with to find the optimal model which avoided overfitting to the training data. The accuracy and loss data over 200 epochs is shown in Figure 9 for 10 hidden layers. The training and validation results suggest that the L2 regularization should be set to $7.5 \times 10^{-4}$ and the dropout

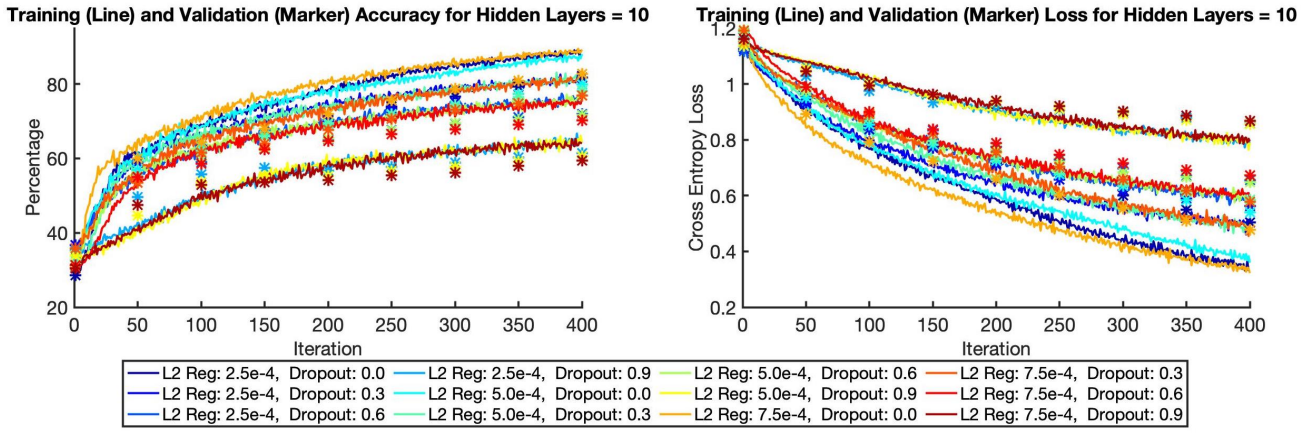


**Figure 9:** Accuracy and loss for validation and training with 10 hidden units. The number of epochs was 200 and the training to test ratio was 0.8.

**Table 1:** Training and validation results after 200 epochs with L2 Regularization of $2.5\times10^{-4}$ and no dropout.

| Number of Hidden Layers | Training Accuracy | Validation Accuracy | Training Loss | Validation Loss |
|---|---|---|---|---|
| **3** | 75.2841 | 72.0581 | 0.6740 | 0.7058 |
| **6** | 85.3456 | 77.8030 | 0.4485 | 0.5922 |
| **9** | 89.0309 | 81.7172 | 0.3322 | 0.5043 |
| **12** | 89.3860 | 84.2803 | 0.3201 | 0.4633 |
| **15** | 91.4062 | 85.7702 | 0.2483 | 0.4241 |
| **18** | 92.2348 | 86.4141 | 0.2167 | 0.4024 |
| **21** | 92.8346 | 85.9343 | 0.2081 | 0.4206 |
| **24** | 93.4975 | 85.3283 | 0.1892 | 0.4439 |

probability be set to 0.0 to maximize training accuracy whilst maintaining low validation and training losses.

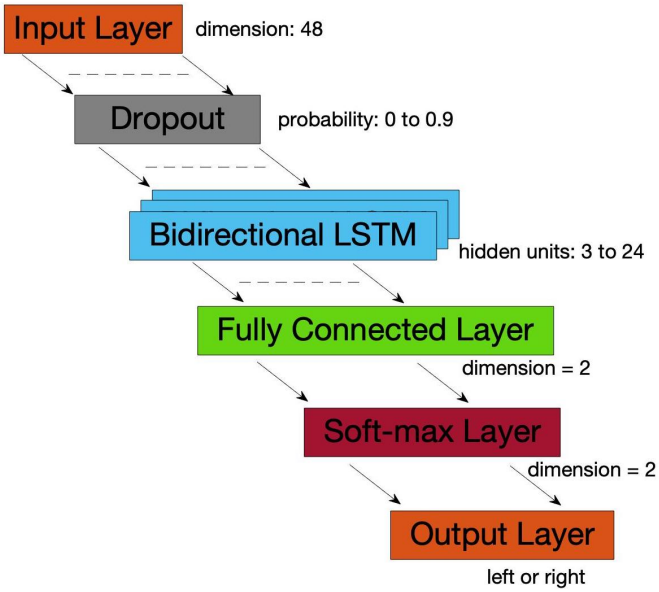


**Figure 10:** Outline of neural network architecture.

The final model was trained using 500 epochs such that the validation loss was approximately stationary.

## Results

The final model achieved a validation accuracy of 89.74% after 500 epochs. Due to inaccuracies in the training data and an imbalance between examples of frames containing motion and those which were stationary, the majority of the errors occurred during frames which depicted the hand being stationary or not present. One hypothesizes these errors were caused by the 2 second pause between changes in motion which caused large differences in sequential frames moving to and from frames which were labelled as stationary. To overcome this, a constant threshold was set to detect stationary frames and frames without the hand present. These thresholds were set empirically using the block integral values shown in Figure 6. The performance of the system was measured on the number of misclassified motion frames out of 2000 test frames. The error rate for the test set was 98.4%.

## Conclusion and Future Work

This work has shown that the 1D array representation is sufficient to classify the direction of rotational motion from an image recorded from inexpensive hardware. Future work can build on this by training to a more diverse range of hand motions. In its current form the system is unable to detect the location of each individual digits of the hand. This can be rectified by reducing the smoothing on the intensity map. In addition, the current system is only able to detect a single hand. As the system is able to detect the presence of hands in a video, block processing could be used to detect the motion of hands in multiple blocks of the screen.

## References

[1] Jamie Schiel, Dr. Andrew Bainbridge-Smith. (2015). Efficient Edge Detection on Low-Cost FPGAs . Retrieved from https://arxiv.org/pdf/1512.00504.pdf

[2] James W. Cooley, Peter A. W. Lewis et al. (1969). The Fast Fourier Transform and Its Applications. Retrieved from http://www.lcs.poli.usp.br/ maria/FFTandApp.pdf

[3] Hojjat Salehinejad, Sharan Sankar et al. (2015). Recent Advances in Recurrent Neural Networks . Retrieved from https://arxiv.org/pdf/1801.01078.pdf

* All code for this project is available at: https://github.com/fpgamy/6344