# INTRODUCTION

Parallel processing has been getting increasing attention lately online, and with good reason. As CPU manufacturers start adding more and more cores to their processors, creating parallel code is a great way to improve performance.

The objective of this project is to implement real time face recognition algorithms, where the images have been taken from a webcam, using parallel processing. This is implemented using OpenCV library, which has the following three algorithms for face recognition:

1. **Eigenfaces**: Eigenfaces refers to an appearance-based approach to face recognition that seeks to capture the variation in a collection of face images and use this information to encode and compare images of individual faces in a holistic (as opposed to a parts-based or feature-based) manner.
2. **Fisherfaces**: Face-like two-dimensional representation of basis vectors obtained by Linear (Fisher's) Discriminant Analysis on set of facial images. When LDA is used to find the subspace representation of a set of face images, the resulting basis vectors defining that space are known as Fisherfaces.
3. **Local Binary Patterns Histograms (LBPH)**: The LBPH method takes a different approach than the eigenfaces method. In LBPH each images is analyzed independently, while the eigenfaces method looks at the dataset as a whole. The LBPH method is somewhat simpler, in the sense that we characterize each image in the dataset locally; and when a new unknown image is provided, we perform the same analysis on it and compare the result to each of the images in the dataset.

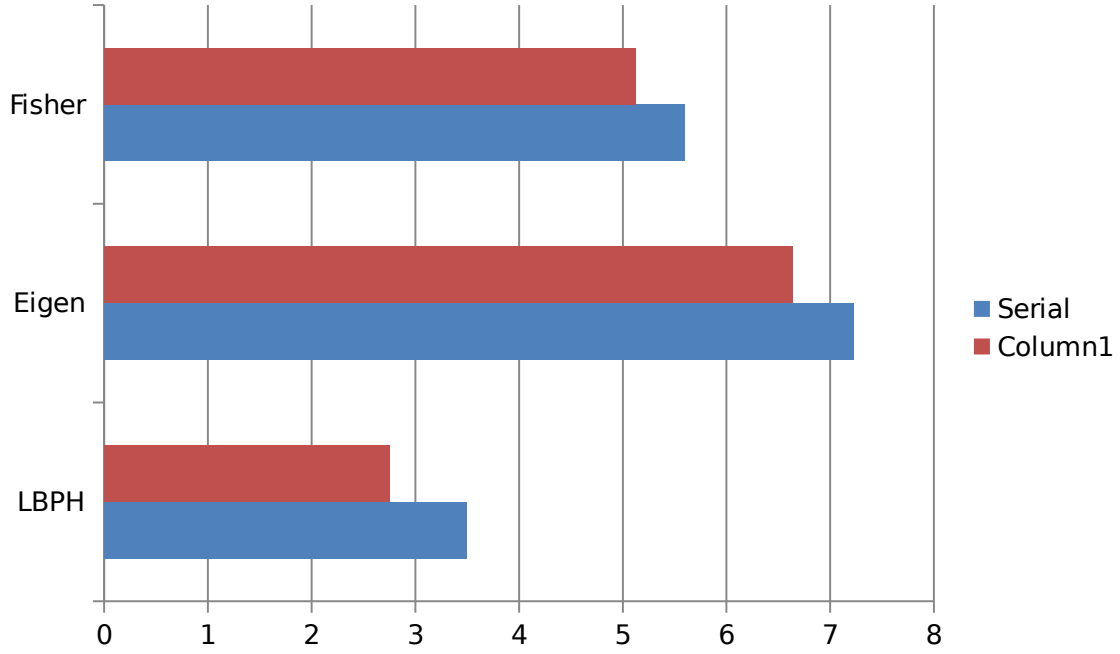
Firstly, the above mentioned face recognition algorithms have been implemented serially. Secondly, the serial algorithms have been redefined so that they can be processed in parallel. Finally, it is demonstrated how parallel processing can accelerate these multi-face recognition algorithms and reduce the processing time as compared to their serial implementations.

We use python's multiprocessing module to implement parallelism. It is a package that supports

spawning processes using an API. The multiprocessing package offers both local and remote concurrency, effectively side-stepping the Global Interpreter Lock by using subprocesses instead of threads. Due to this, the multiprocessing module allows the programmer to fully leverage multiple processors on a given machine.

The multiprocessing module also introduces APIs which do not have analogs in the threading module. A prime example of this is the Pool object which offers a convenient means of parallelizing the execution of a function across multiple input values, distributing the input data across processes (data parallelism).

# RESULTS AND DISCUSSION




We see that parallel implementation of all the face recognition algorithms is faster than the serial implementations by

1. LBPH – 1.26 times
2. Fisher – 1.09 times
3. Eigen – 1.08 times

# CONCLUSION

From the results above, we conclude that parallel implementation of all the face recognition algorithms are faster than their corresponding serial implementations. Also, we see that LBPH Face Recognition algorithm is faster than Eigen and Fisher Face Recognition algorithm in general.

# REFERENCES

[1] OpenCV Face Recognition Tutorial
“http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html”
[2] Multiprocessing Python Documentation https://docs.python.org/2/library/multiprocessing.html
[3] Face Recognition using Python and OpenCV “http://hanzratech.in/2015/02/03/face-recognition-using-opencv.html”