**Introduction:**

The performance of popular Optical Character Recognition (OCR) engines such as Tesseract can be greatly improved by pre-processing of the image. When texts exist in an image alongside with other objects, OCR engines often return random characters and carriage returns. To combat this problem ITSegmenter.js detect and isolate text regions in an image.

The first step to processing an image is to decode it into pixel data, thankfully that is a built-in feature of the image element. To access the pixel data, the image is drawn on a canvas element, it can then be accessed by the getImageData method. From there a convolution matrix filter is applied to the image data to sharpen it. This improves the results of the next step, feature detection. After sharpening the image, Features from Accelerated Segment Test (FAST) is used to find the corners in the image. Then the corners coordinates stored in an array are used to construct a k-dimensional tree. From there, the Density-based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is applied to corners, which groups them together. Once the clusters are formed, bounding boxes can be drawn based on the maximum and minimum x/y values of each cluster.

Getting started:

1. Include the following in the head of the html:

'<script src="src\ITSegmenter.js"></script>'

1. textSegment(imgPath);
2. The coordinates of the bounding box for each cluster will be stored at outputRects {}
3. To paint the output directly onto a canvas specify the canvas element id in the parameters  
   textSegment(imgPath, fThreshhold, eps, minPts, sharpness, drawRects, splitRects, convertToImage, canvasId)

**ITSegmenter.js**

Convolution Matrix Filter:

DBSCAN:

FAST:

K-dimensional tree: