Line Recognition for Generating Accessible Line Plots

Line recognition for generating accessible line plots

Francesco Lombardi
University of Florence
Department of Information Engineering
Florence, Italy
Email: francesco.lombardi3@stud.unifi.it

Cagatay Goncu
Monash University
Melbourne, Australia
Email: cagatay.goncu@monash.edu

Simone Marinai
University of Florence
Department of Information Engineering
Florence, Italy
Email: simone.marinai@unifi.it

***Abstract*—Existing technologies identify individual lines in a line plot by matching the symbols in the legend. Identifying each symbol is crucial to create accessible graphics for people who are blind so that they provide the best resemblance possible to the original plot. This resemblance provides better communication between sighted and vision impaired users. In our work we use an existing network architecture and improve it in a way that it will get more semantic information out of the lines and their symbology. We present these line graphs in the GraVVITAS system, an approach for providing dynamic and refreshable accessible graphics, to be used by people who are blind.**

I. Introduction

Line graphs are one of the most commonly used graphs in published media and in scientific literature. However, they are not easily accessible by people with vision impairment. One of the major issues which causes this lack of accessibility is that their semantic is not part of the graphic production [4]. The visual components are initially created by using various software packages and then these graphics are manually edited to make it suitable for the final media. During this process all the semantic information are lost and the line graph is often saved in a still image.

We recognise the hurdles of creating such accessible graphics for print and digital media from the graphic designers perspective. Instead of trying to capture all the semantics during the design process, we investigated the recognition of these semantics from the original graphics after they are produced.

In this paper, we propose a system which takes a line graph as an input and recognises the semantic information. These information are then added to a symbolic representation of the graphic so that they can be presented in various formats. In this paper, we explored their use with the GraVVITAS system that we already adopted to present floor plans to visually impaired users [3].

GraVVITAS is a practical, generic and low cost solution for providing dynamic and refreshable accessible graphics. It is used to create and present graphics by using multi-touch screens of standard tablets augmented with haptic and audio feedback [5].

II. Background

Existing research has focused on solutions that split the large and complex task of line plot analysis into several subproblems such as recognition of cartesian axes, legends and individual lines [9]. They also focus on classification of the types of existing graphs in scientific documents, like line plots, bar charts, scatter plots [12] [2].

In this paper we apply image processing and deep learning techniques to the problem of line detection within line plots. We tackled the problem by looking at it from a different point of view than [8] [10]. In [8] authors maintain an approach strongly linked to pure image analysis techniques and mainly orient their study towards the search for a path between the pixels of the same curve, applying traditional techniques such as PCC (Primitive Chain Code), while [10] mainly aims to interpret information contained in scientific papers plots. In FigureSeer, instead, authors investigate to parse the graphic content in research papers to improve the search and retrieval of information [11]. Their approach is to create a unified process of graphical component detection including the axes, legend and individual lines.

In our work we adopt the approach in FigureSeer and extend it with specific considerations for accessible line plot generation.

III. Design

FigureSeer identifies individual lines in a line plot by matching the symbols in the legend. This is helpful to create an equivalent representation of the line plot in another format. However, identifying each symbol is also crucial to create accessible graphics that provides the best resemblance possible to the original graph. This resemblance provides better communication between sighted and vision impaired users as they can refer to the lines by using the same visual properties such as colour and style.

In our work we use the network architecture proposed in [11] as our baseline and improve it in a way that it will be possible to get more semantic information out of the lines and their symbology. We use these additional information to present the graphs in GraVVITAS.

Our work is an ongoing work and at this stage we have implemented the line detection module which finds the individual lines in a line graph and outputs their coordinate points as well as other associated information given on the legend of the graph in a format that is compatible with the GraVVITAS system. We will be extending the system so that it will also include the axis detection and legend detection

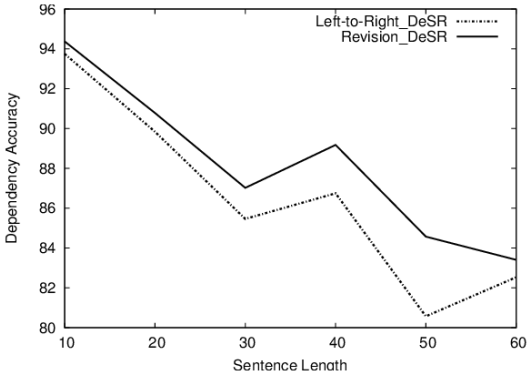


Figure 1: One sample line graph.

and understanding module specifically aiming for a complete representation on the GraVVITAS system.

A. Line Detection Module

Following [11] we investigated two approaches to achieve better line detection: (i) one image processing approach, and (ii) one CNN based approach. Figure 1 shows a sample test image from the dataset [1].

1) Image Processing Approach: in this case we consider the following steps to detect the lines inside one line graph:

- From the dataset we used, we know a set of annotated color images of line plots and the corresponding legends and legend symbols positions;
- We crop all the plot areas and symbols in the legend;
- In each symbols crop, we detect the main color (the color of the relative curve in the plot) using color clustering technique (k-means with 2 clusters for clustering);
- Once the main color has been detected, we use a threshold triple of values (for R, G, B) to set the interval of values in which a pixel is said to belong to a curve.

In this way every color difference in graphs can be detected, however several images in the dataset are black-and-white or contain different lines of the same color but different line patterns.

In order to solve this problem, similarly to [11], we have created a feature map f for each line in each image by taking each symbol crop and rotationally convolving it over the image, having as main objective to make some kind of template matching of the crop into the plot area. Being convolution a rotation-invariant operation, we convolved 30 to 60 different rotations of each crop with each plot, and we finally created each feature map by taking the maxima of the resulting tensor for each image pixel. It is worth observing here that even if the convolution is a part of deep learning models, in this case this operator is performed in the classical image processing way.

This approach has brought some improvements, but its application to the huge variety of different symbol patterns across the images we considered has given variable results. As we can notice from the images in Figure 2 and Figure 3, solid lines can be constructed using the intensity levels of the output

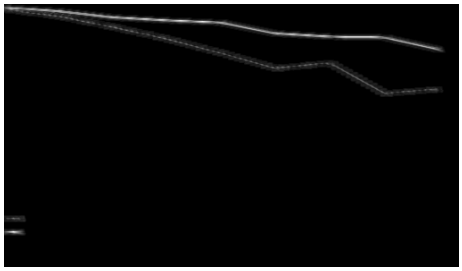


Figure 2: Output image from applying a solid line filter to the original image

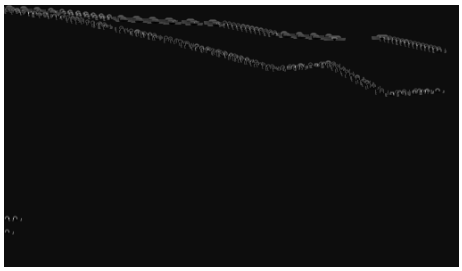


Figure 3: Output image from applying a dashed line filter to the original image

image, however it is not possible to achieve similar results for the dashed lines. Furthermore this approach does not provide good results for line styles that have different patterns such as squares and circles (like e.g. $-o-$ or $-\Diamond-$) as shown in Figure 4. We therefore decided to take a different approach which is based on a Convolutional Neural Networks.

2) CNN-Based Approach: we use a CNN-based approach [6] [13] to derive feature maps similar to those previously defined, which allows us to implicitly define and model our patch transformation (rotations, overlapping, etc.). Similarly to [11], we learn the embedding of images patches to a low dimensional features vector using a Siamese Neural Network based on [13]. Basically, the Siamese Network learns the similarity between the line in the legend with lines in the plot. Once trained, the network can then be used to detect the lines matching each item in the legend. Taking again as input the plot in Figure 1 the similarity scores computed by the Siamese network for each of the two plots are shown in Figure 5. With

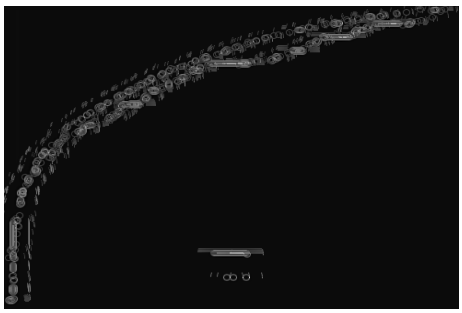


Figure 4: Output image from applying a pattern filter to the original image.

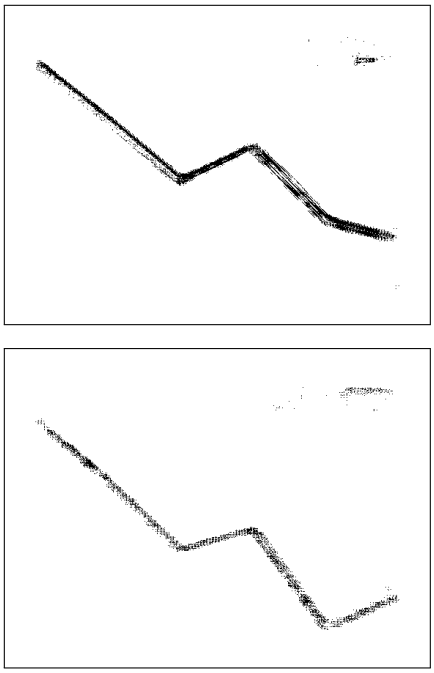


Figure 5: Feature maps generated by the CNN-based approach. Each line in Figure 1 is represented in a separate image.

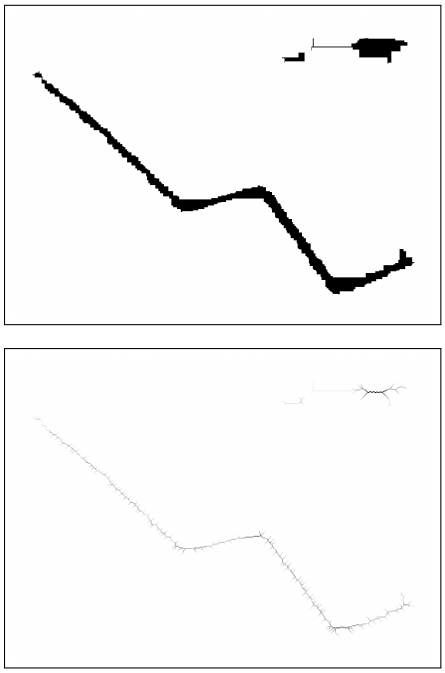


Figure 6: Processing of the feature map obtained by the CNN-based approach. Top: closing of the feature map (dilation followed by erosion with the same structuring element). Bottom: medial axis computation.

respect to the previous approach, the density of the feature map computed by the Siamese network provides better recognition of lines in terms of their styles. As we can notice, also the part of the image corresponding to the legend looks similar to itself and is therefore highlighted in the feature map (on the top-right corner of the image). In the prediction step the output of the Siamese network is computed with a sliding window with stride of two pixels. We trained the Siamese network using a Nvidia GTX-970 GPU for around 10 hours. Testing the model less than an hour for 200 images.

After generating with the Siamese network the feature map we need to extract a symbolic representation of the plot. To this purpose, we first consider some morphological operations to extract the skeleton of the line and then consider a dynamic programming approach to detect the line path from the corresponding feature maps.

B. Accessible content creation module

We process the feature maps computed from the CNN-based approach with a morphological erosion and dilation with a 5×5 pixels kernel followed by medial axis computation. The intermediate steps for the bottom feature map in Figure 5 are shown in Figure 6. From top to bottom we show the closed image (obtained by first dilating and then eroding the image) and the skeleton obtained by the medial axis computation. As we can notice the skeleton contains several artifacts and it is of course computed also for the example line belonging to the legend.

The resulting medial axis represents a good approximation of the line path, but cannot be directly used to provide a symbolic representation of the line. In particular, we should obtain a list of points to be inserted as a polyline in the SVG file that is required as input to the GraVVITAS application.

To generate this list of points one simple approach is to sample the points in the feature map at regularly spaced positions in the horizontal axis and then emit one output value for each of these samples on the basis of the skeleton. One disadvantage of this approach is the relatively low precision, since also artifacts can be sampled instead of the "true" skeleton. Another problem is the potential presence of points coming from the legend in the output list.

To avoid these risks the output is processed by a dynamic programming algorithm to infer the real curve path. In particular, similarly to that in [11], our goal is to search for the optimal path $P_s = \{p_i\}_{i=1}^{n} = \{(x_i, y_i)\}_{i=1}^{n}$ which is computed by optimizing the following function:

$$E(s) = \sum_{i=1}^{n} \alpha \cdot f(p_i) + \sum_{i=1, j=i+1}^{n-1} \beta \cdot f(p_i, p_j) \qquad (1)$$

$$t.c., \forall i, 1 \leq y_i \leq m, 1 \leq x_i \leq n, x_{i+1} = x_i + 1$$

Where the first term indicates the likelihood of a pixel p_i belonging to the path, given its feature map f, while the second is used to "encourage" smooth transitions between two adjacent pixels penalizing the contribution given by pixels to values of y which are distant from each other.

The output of this algorithm, shown in Figure 7, is used to create an SVG file, input for GraVVITAS. By closely inspecting the figure, we can notice that there is no output for the line in the legend in the top-right part of the image. However, the

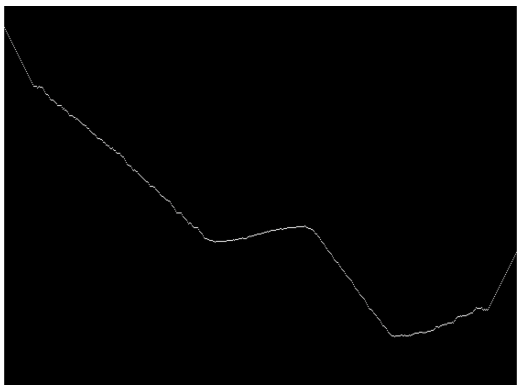


Figure 7: Dynamic programming algorithm output for the plot in Figure 1.

algorithm produces one output (actually a straight segment) also for the first and last part of the plot that is not present in the original line. These parts can be easily removed by exporting in the SVG file only points intersecting with the closed feature map (top map in Figure 6).

C. SVG generation

The GraVVITAS system accepts an SVG file to present accessible graphics. Thus, our system outputs the resulting SVG in the following format where each polyline represents a line in the original image (for simplicity we omit several details and in particular actual coordinates of the polylines). In the example, we first define the vertical and horizontal axes with the corresponding positions. The next two polylines define the two lines in the example on Figure 1; one dashed and one solid line. The second part of the SVG contains information about haptic and audio feedback that can be differentiated for each line. In this case the attributes of SVG elements are not set to any value and get the default.

```
<svg id="svg-graphic" xmlns="" xmlns:xlink=""
width="" height="" viewBox=""
preserveAspectRatio="">

    <title></title>

    <g id="" transform="">
        <polyline points="0 0 0 800" id="Y axis"
        class="" style="">
                <title>Y axis</title>
        </polyline>

        <polyline points="0 800 1200 800"
        id="X axis" class="" style="">
                <title>X axis</title>
        </polyline>

        <polyline
        points="...."
        id="line1"
        class="" style="">
                <title>Dashed line</title>
        </polyline>

        <polyline
        points="...."
        id="line2"
        class="" style="">
                <title>Solid line</title>
        </polyline>
    </g>

<metadata id="" rpid="" title="" description=""
category="" group="" subgroup="" keywords=""
collections="" orientation=""
targetdevicename="">

        <summary></summary>
        <gravvitas ng-repeat="">
            <id>line1</id>
            <interiorcolor></interiorcolor>
            <bordercolor></bordercolor>
            <cornercolor></cornercolor>
            <audio></audio>
            <volume></volume>
            <text>Dashed line</text>
            <vibration></vibration>
            <annotation></annotation>
        </gravvitas>

        <gravvitas ng-repeat="">
            <id>line2</id>
            <interiorcolor></interiorcolor>
            <bordercolor></bordercolor>
            <cornercolor></cornercolor>
            <audio></audio>
            <volume></volume>
            <text>Solid line</text>
            <vibration></vibration>
            <annotation></annotation>
        </gravvitas>

 </metadata>
</svg>
```

IV. System Test

A correct evaluation of a visualization tool of graphical information for visually impaired people would require a suitable usability test by final users. In this case this would require visually impaired people to use the system to inspect the line plots and understand its content (e.g inferring the mutual relationships among lines). As a preliminary partial evaluation we show in Figure 8 three input plots with different features and the corresponding output visualized with the GraVVITAS system on an iPad. Different lines are shown enlarged and with false colors. Thicker lines (with respect to the input image) are required because the spatial resolution on fingers is smaller than the visual acuity. In other words if the lines are too thin it is really difficult, if not impossible, to follow them by using the audio feedback provided by the GraVVITAS system since with a small displacement we would miss the line. The false colors are of course not visible to visually impaired people. Rather, these are used by the

GraVVITAS system to provide a different audio feedback for each line.

Given these considerations by inspecting Figure 8 we can notice that the conversion system can deal with different types on line plots handling black and white and colored plots as well as more than two lines.

As a support to what has been presented, we also show some quantitative results. Each point p_i belonging to the predicted path $P_s = \{p_i\}_{i=1}^{n} = \{(x_i, y_i)\}_{i=1}^{n}$ is counted as a true positive if the difference with the ground-truth y_i' is below a threshold *t*, i.e $(y - y_i') \leq t$ [11] (*t* = 2% image height in our experiments). A predicted point is counted as false positive if there exists no curve point in the corresponding ground-truth image, while a false negative is assessed as such if the ground-truth image contains a non-predicted point. Moreover, if the distance between the real y value and the predicted one is above the threshold *t* both a false positive and a false negative are counted up.

With regard to these definitions, the measure we chose as the evaluation metric is the F-measure [7]. We considered a line path correctly predicted if its F1-score value is higher than 95%. The F1-scores of the evaluated images range from 97% to 99%, and this high accuracy is clearly shown in Figure 8. Note that several components need to be sequentially accurate for the entire parsing to be considered correct.

V. Conclusions and Future Work

In this paper we presented our current work on the conversion of lines plots for the visualization to visually impaired people. We demonstrated a full pipeline from the line plot image into a visualization in the GraVVITAS system that can be used to inspect and understand the plots.

In the ongoing work we will be further developing the CNN architecture for line plot recognition. This process will include investigations on existing systems and determining whether they can be used to improve our system which is specifically targeting to create accessible line plots.

Acknowledgment

Cagatay Goncu is supported by the Australian Research Council (ARC) grant DE180100057. He is a co-founder at RaisedPixels Pty. Ltd. (www.raisedpixels.com) which is commercialising and maintaining GraVVITAS on Apple App Store under the name Raised Pixels Reader App.

References

[1] Allen Institute for Artificial Intelligence. *Figureseer dataset*. https://allenai.org/plato/figureseer/.

[2] Abhijit Balaji, Thuvaarakkesh Ramanathan, and Venkateshwarlu Sonathi. "Chart-Text: A Fully Automated Chart Image Descriptor". In: *arXiv preprint arXiv:1812.10636* (2018).

[3] Cagatay Goncu, Anuradha Madugalla, Simone Marinai, and Kim Marriott. "Accessible On-Line Floor Plans". In: *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*. 2015, pp. 388–398. DOI: 10.1145/2736277.2741660.

[4] Cagatay Goncu, Simone Marinai, and Kim Marriott. "Generation of accessible graphics". In: *22nd Mediterranean Conference on Control and Automation, Palermo, Italy, June 16-19, 2014*. 2014, pp. 169–174. DOI: 10.1109/MED.2014.6961366.

[5] Cagatay Goncu and Kim Marriott. "GraVVITAS: generic multi-touch presentation of accessible graphics". In: *IFIP Conference on Human-Computer Interaction*. Springer. 2011, pp. 30–48.

[6] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. "Matchnet: Unifying feature and metric learning for patch-based matching". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3279–3286.

[7] Xiaodi Hou, Alan Yuille, and Christof Koch. "Boundary detection benchmarking: Beyond f-measures". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2123–2130.

[8] Xiaonan Lu, J Wang, Prasenjit Mitra, and C Lee Giles. "Automatic extraction of data from 2-d plots in documents". In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 1. IEEE. 2007, pp. 188–192.

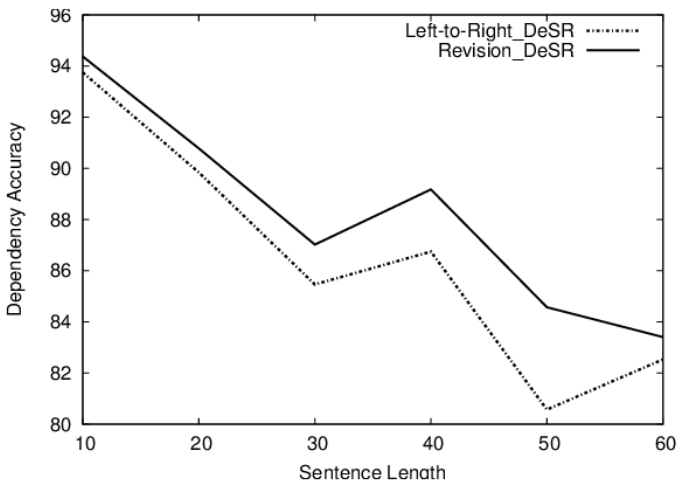
[9] Rathin Radhakrishnan Nair, Nishant Sankaran, Ifeoma Nwogu, and Venu Govindaraju. "Automated analysis of line plots in documents". In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE. 2015, pp. 796–800.

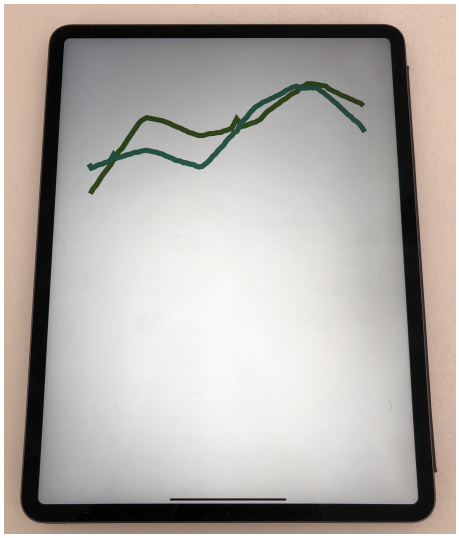
[10] Rathin Radhakrishnan Nair, Nishant Sankaran, Ifeoma Nwogu, and Venu Govindaraju. "Understanding line plots using Bayesian Network". In: *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. IEEE. 2016, pp. 108–113.

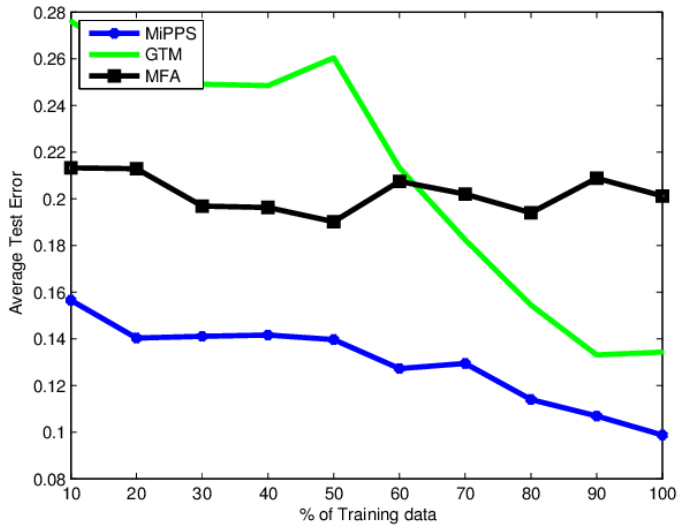
[11] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi. "FigureSeer: Parsing result-figures in research papers". In: *European Conference on Computer Vision*. Springer. 2016, pp. 664–680.

[12] Binbin Tang, Xiao Liu, Jie Lei, Mingli Song, Dapeng Tao, Shuifa Sun, and Fangmin Dong. "Deepchart: Combining deep convolutional networks and deep belief networks in chart classification". In: *Signal Processing* 124 (2016), pp. 156–161.

[13] Sergey Zagoruyko and Nikos Komodakis. "Learning to compare image patches via convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 4353–4361.







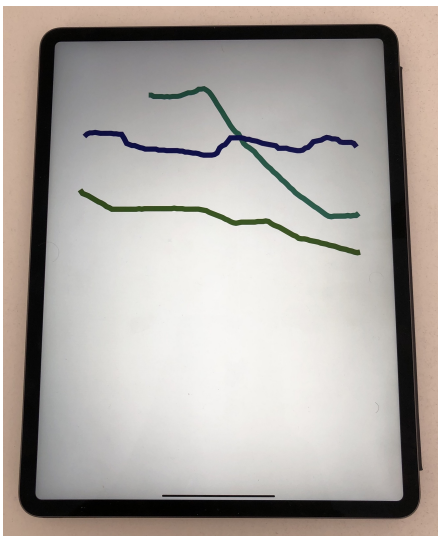


Figure 8: Three line plots and the corresponding output in the GraVVITAS system.