Chang Han (u1472415), Ishrat Jahan Eliza (u1472593)

GitHub Link: https://github.com/elizaan/Color-to-Pattern.git

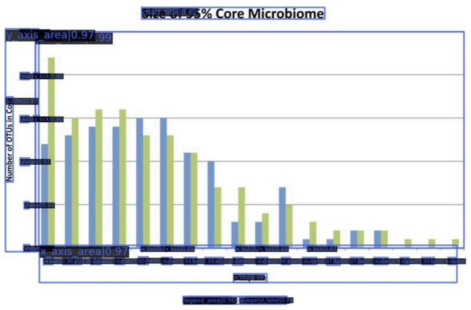
Motivation

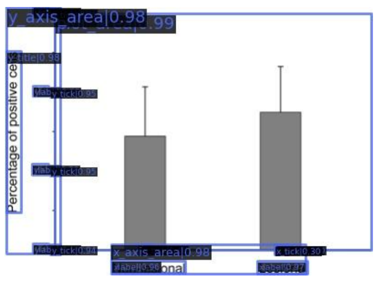
Textures in visualization design have been recognized as an accessible alternative to colors. They are helpful for color-blind individuals and useful for low-color displays (such as e-ink) and physical printouts. However, despite their proven effectiveness and the design space exploration, textures are still rarely used in practice and likely cannot (and perhaps should not) replace the dominant role of color in visualization design. To bridge this gap, we propose a tool to convert the colored visualizations into textured visualizations. The main purpose is to reduce the cost of manual editing and make the process as automated as possible. Machine learning workflow can make the process of chart area and data type detections automatic and directly feed the extracted information into the texture chart generation workflow.

Our Solution and Experimental Design

Chart Elements Detection

To identify the colored part and data types and to convert the plot images later to generate texture plots, we need to separate the regions of interest. In our case, the region of interest is only the plot area, but there are no other plot elements, such as titles, ticks, labels, legends, etc.

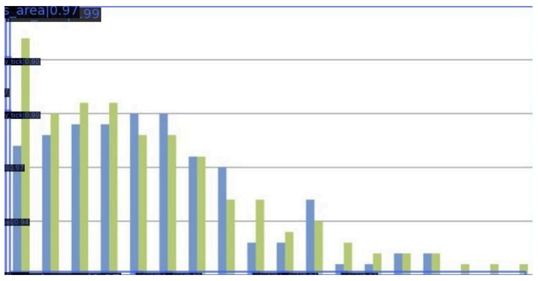


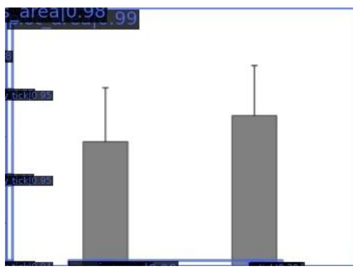


We detect the chart/ plot elements in this project using the CACHED (Context-Aware Chart Element Detection) [1] model. We used the R-CNN framework to integrate a local-global context fusion module consisting of visual context enhancement and positional context encoding from CACHED. The final model is called the cascade R-CNN framework. The model classifies 18 chart elements, such as x_title, y_title, x_ticks, y_ticks, plot_area, xlabel, ylabel, x_axis_area, y_axis_area, tick_grouping, chart_title, etc. and detects them. The attached images are two examples of the final result of chart element detection using the Cascade R-CNN model. The dataset we used here is from the PubMed Central (PMC) Chart Dataset that was released and updated with the Chart Competition in ICDAR 2019 [5] and ICPR 2022 [6].

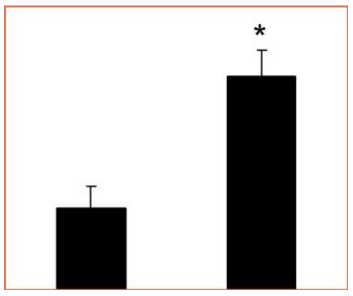
The Cascade R-CNN model only detects the chart element but doesn't generate separate images for each component from the parent chart. We need the plot area as input for the second part of our project. Thus, we enhanced the method to fulfill the purpose. We used the

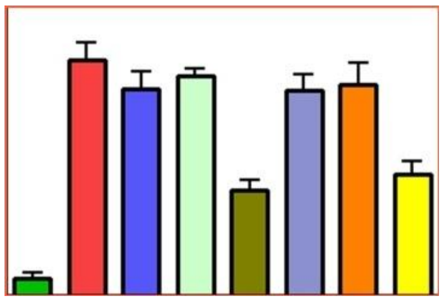
Chang Han (u1472415), Ishrat Jahan Eliza (u1472593)

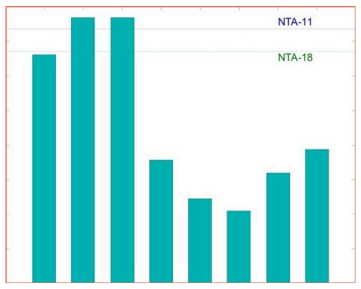




OpenCV module in Python to read the image that detected areas in bounding boxes. The openCV converts the image into a numpy array and extracts the region of interest, i.e., the plot area in our case, using the array slicing techniques. Later, we modified this model to get a cleaner plot area to minimize residues and junk that might create distractions in segmenting the plot area. Below are some final plot areas extracted from our modified model after removing the labels, bounding-box overlaps, and other junk. The IoU (Intersection over union) metric measured that the model is around 89.35 % well predicted the bounding box. The score_a metric returns a value of 93.75%.







Color-data Type Detection

For the second part of our project, we aim to **(i) identify the different colored parts of visualizations** and (ii) **determine the data types that the colors are encoding**. We began by exploring existing tools. ChartSense [3] uses an interactive approach to extract data from visualizations, while ChartOCR [2] goes further with fully automated data extraction. However, both methods face accuracy issues—ChartOCR has a mean error of around 0.1, and ChartSense requires significant manual effort to correct the data. Additionally, these approaches rely heavily on training corpora and manually set rules, limiting their effectiveness to specific charts. This is not ideal for our application, which aims to support a wider variety of chart types.

Fortunately, unlike previous methods, we don't need to extract precise data values but rather identify 'data patterns' (quantitative, ordinal, or categorical) and segment the different colored

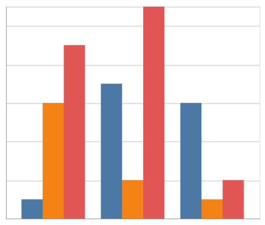
$$\Delta E_{00} = \sqrt{\left(\frac{\Delta L'}{k_L S_L}\right)^2 + \left(\frac{\Delta C'}{k_C S_C}\right)^2 + \left(\frac{\Delta H'}{k_H S_H}\right)^2 + R_T \frac{\Delta C' \Delta H'}{S_C S_H}}$$

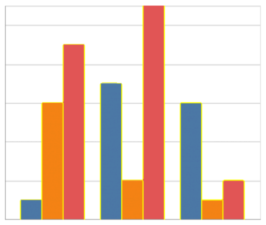
where:

- $\Delta L'$, $\Delta C'$, $\Delta H'$ are the adjusted differences in lightness, chroma, and hue.
- S_L, S_C, S_H are scaling factors.
- R_T is a rotation term that accounts for interactions between chroma and hue differences.

parts. To address this, we employed a simplified approach. We first detected the boundaries of the colored areas using the SLIC superpixel algorithm, recognizing that some charts use continuous color gradients. To handle this, we set a threshold and apply Gaussian blur to classify these charts appropriately.

Next, we transformed the colors into the **CIELAB** color space, which aligns more closely with human visual perception. In CIELAB, equal color differences correspond more accurately to perceived differences compared to RGB and HSV. We then used the CIEDE2000 Color Difference Formula [4] (shown on the left) to calculate the differences between adjacent colors. If the difference exceeds a threshold L, we classify it as categorical data; if the difference is smaller, we classify it as quantitative or ordinal data.


original image


segments with fluorescent boundary

mean distance is 20.9 > 10

categorical

Texture (Pattern) Design

We explored the design space of textures on visualization charts and proposed a (black-and-white) texture palette. The framework heavily relies on the perceptual studies done by He et al.[7], where we outline a systematic approach to designing a black-and-white texture palette that effectively replaces a categorical color palette. We implemented a naive realization of the palette as a preliminary approach for the target—an automatic transition from colored charts to textured charts. Here, we focus on two distinct categories of color (texture) encoding: 1. Categorical, 2. Quantitative. The former is especially important as quantitative color-encoding can still show information to some extent with black-and-white applications, but categorical color-encoding is usually lost.

Categorical Texture Palette

Design Goals

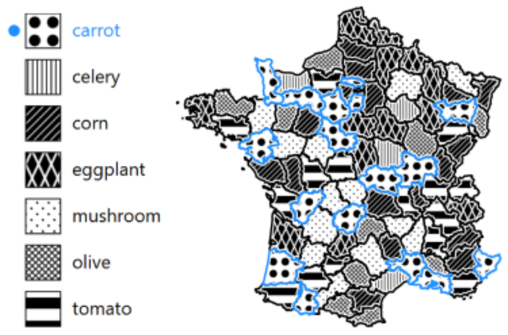
The framework by He et al. emphasizes two key types of textures: **geometric** (abstract shapes) and **iconic** (representative symbols). A successful texture palette depends on careful manipulation of attributes like **density**, **size**, **orientation**, **randomness**, and **background color** to create distinctive yet cohesive patterns. Based on that, we identified four design goals for our texture palette.

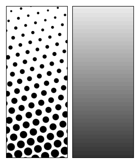
Chang Han (u1472415), Ishrat Jahan Eliza (u1472593)

1. Clarity: Each texture must be easily distinguishable, even at small scales or low resolutions.
2. Aesthetic Harmony: Textures should complement the overall design of the visualization and avoid overwhelming the user.
3. Accessibility: Consider users with visual impairments, ensuring textures are legible in tactile or embossed formats.
4. Easy Recognition: There should be sufficient distinction between different textures, making them easily recognizable at a glance.

Base Texture Types - categorical

We investigated a mix of geometric and iconic textures in existing research literature and commercial tools (e.g., Adobe Illustrator), aiming to maximize variety when designing. Geometric textures are mainly composed of patterns of lines, grids, or dots. For instance, parallel lines can vary in density and orientation, while grids can incorporate angles and spacing adjustments. Iconic textures, on the other hand, are simplified icons relevant to the data categories. Eventually, we designed our palette using the tool provided in Texture Design Interface. **Seen below for an example of seven distinct texture types and an example visualization of a map.**



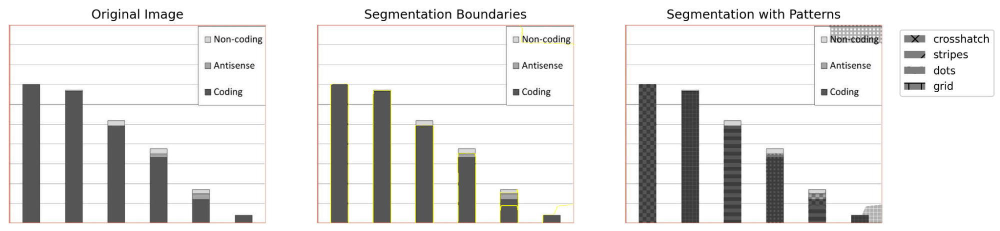


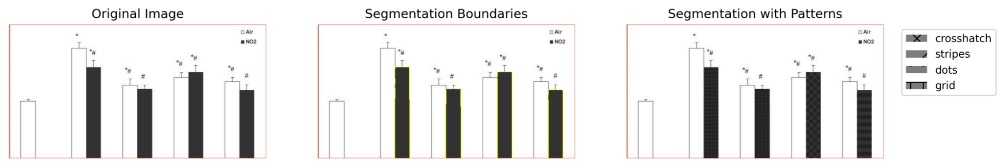
Base Texture Types - additional design for quantitative data.

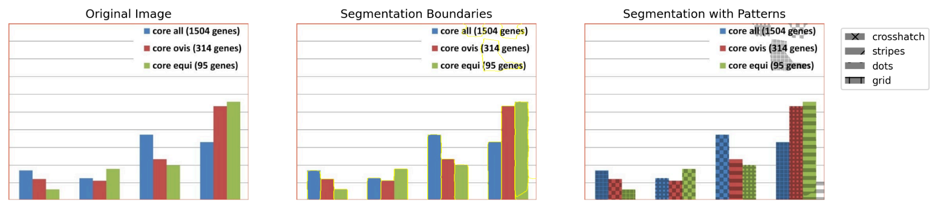
Additionally, we explored the design of textures to encode quantitative data. We discovered a method of reprography called halftone. It uses varied-sized dots to simulate continuous tone, which suits our application. See left for an example from wikipedia. We consider this as a design choice and future work to implement.

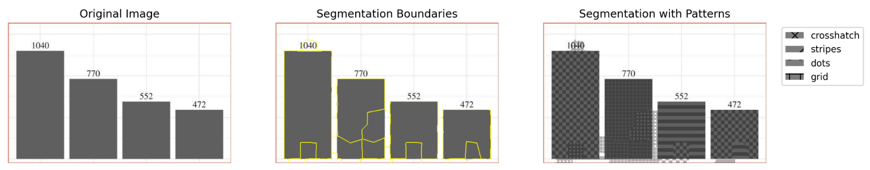
Implementation using ML pipeline

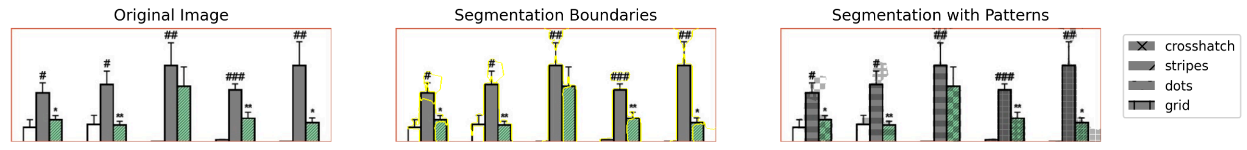
We implement the texture (pattern) for the bar chart dataset mentioned earlier. We used our machine learning workflow to detect plot areas to segment the bar plot components and data types of the components. We mapped the numerical data to the categorical data following our design goals and replaced the segmented bars with the generated textures. In this process, the main challenge has been the proper segmentation of the bars because the other elements like axis ticks axis co-ordinate lines work as distractions, and removing them using proper segmentation logic was a challenge. Below, we add some of the reference outputs that generated textures correctly. The leftmost image is the generated image from part 1 of our project, the middle one is the segmented bars detected and the rightmost part is the textured/ patterned bars we generated.



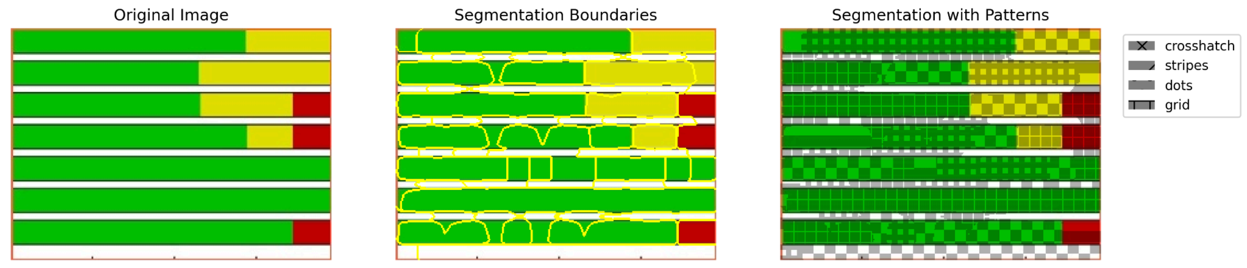








In many cases, when there are stacked bar charts were present, our model hallucinated the bars with other plot marks, and thus the generation texture was misplaced. Below is such a failed example.



Chang Han (u1472415), Ishrat Jahan Eliza (u1472593)

Future Plan

We find the detection boundary, and when the boundary labels generated by the cascade R-CNN model generate overlaps, it still generates some noisy outputs. This noisy region works as a hurdle to detecting individual colored parts of the plot area later. On the other hand, for the stacked bar chart, the detection sometimes is hallucinated. We would like to work on that if we had some more time.
We have generated output based on around 5k images of bar plots. We also want to test the result with other plot types, such as pie charts and scatter plots, to see how it performs for the data types, primarily how the second part of the project (detecting colored parts) works with the current implementation. Additionally, we want to explore other popular datasets used in other chart element detection research work. Another future work would be an expert qualitative evaluation of our designed palettes, as this can not be numerically evaluated. Human evaluation seems to be the reasonable next step.

References

[1] Yan, P., Ahmed, S., Doermann, D. (2023). Context-Aware Chart Element Detection. In: Fink, G.A., Jain, R., Kise, K., Zanibbi, R. (eds) *Document Analysis and Recognition - ICDAR 2023*. Lecture Notes in Computer Science, vol 14187. Springer, Cham. https://doi.org/10.1007/978-3-031-41676-7_13
[2] J. Luo, Z. Li, J. Wang and C. -Y. Lin, "ChartOCR: Data Extraction from Charts Images via a Deep Hybrid Framework," *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, HI, USA, 2021, pp. 1916-1924, doi: 10.1109/WACV48630.2021.00196.
[3] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. 2017. ChartSense: Interactive Data Extraction from Chart Images. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). Association for Computing Machinery, New York, NY, USA, 6706–6717.
[4] Sharma, Gaurav, Wencheng Wu, and Edul N. Dalal. "The CIEDE2000 color‐difference formula: Implementation notes, supplementary test data, and mathematical observations." *Color Research & Application: Endorsed by Inter‐Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur* 30.1 (2005): 21-30.
[5] Davila, K., Tensmeyer, C., Shekhar, S., Singh, H., Setlur, S., Govindaraju, V.: *ICPR 2020 - competition on harvesting raw tables from infographics*. In: Del Bimbo, A., et al. (eds.) ICPR 2021. LNCS, vol. 12668, pp. 361–380. Springer, Cham (2021). doi: 10.1007/978-3-030-68793-9_27.
[6] Davila, K., Xu, F., Ahmed, S., Mendoza, D.A., Setlur, S., Govindaraju, V.: *ICPR 2022: challenge on harvesting raw tables from infographics (chart-infographics)*. In: 2022 26th International Conference on Pattern Recognition (ICPR), pp. 4995–5001. IEEE (2022)
[7] T. He, Y. Zhong, P. Isenberg and T. Isenberg, "Design Characterization for Black-and-White Textures in Visualization" in IEEE Transactions on Visualization & Computer Graphics, vol. 30, no. 01, pp. 1019-1029, Jan. 2024, doi: 10.1109/TVCG.2023.3326941.