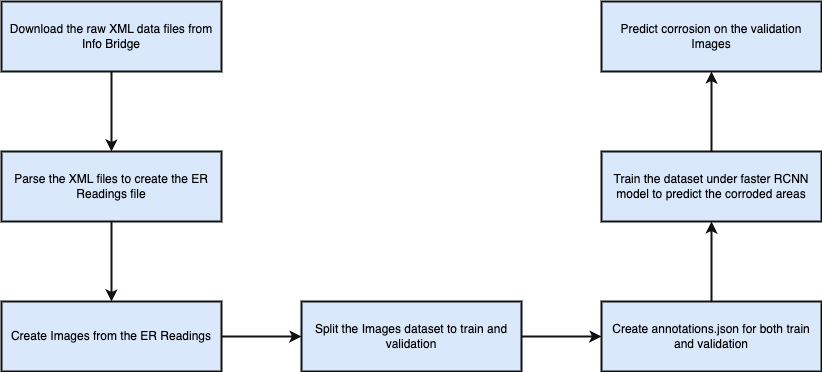
**Electrical Resistivity - (Nondestructive Technique) Evaluation of Concrete Bridge Decks**

**TEAM 3:**

Rishitha Madipelli

**The System Architecture:**

****

**Baseline Solution:**

**Data Collection:** The data for this project was collected from InfoBridge, encompassing 38 bridges. The data consists of XML files, which were parsed to extract Electrical Resistivity (ER) readings. These readings were subsequently used to generate images for further analysis.

**Data Extraction:** Based on the provided guidelines, the ER readings were categorized as follows:

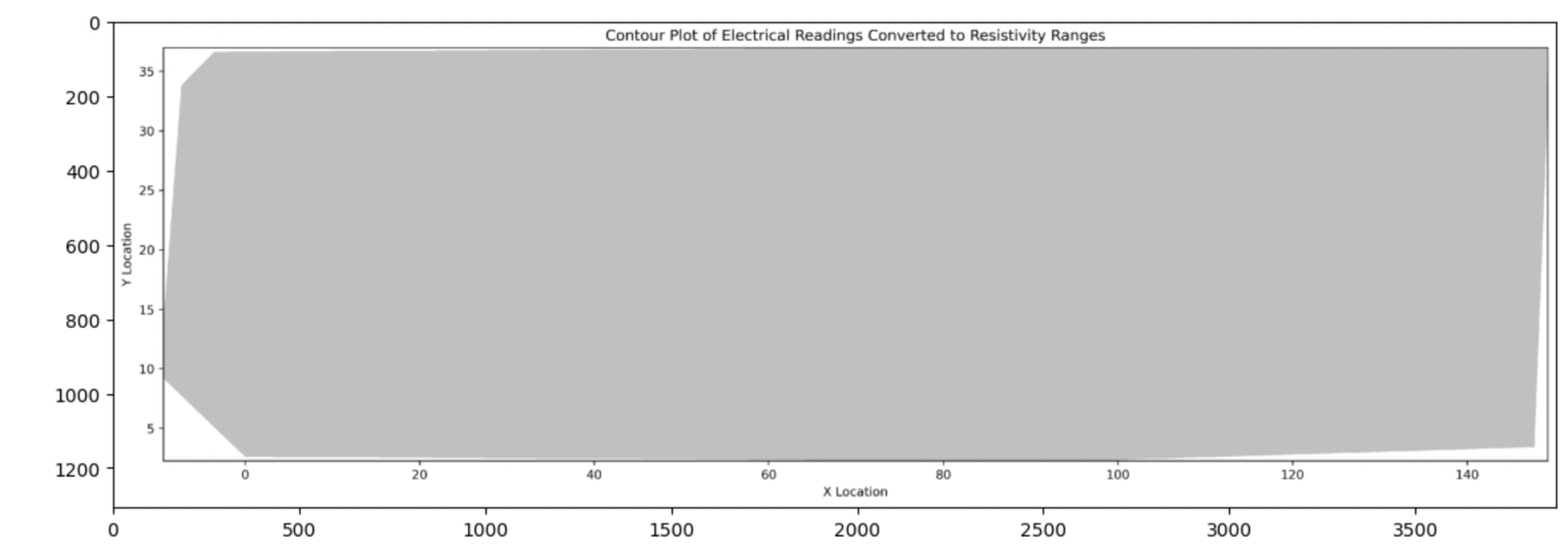
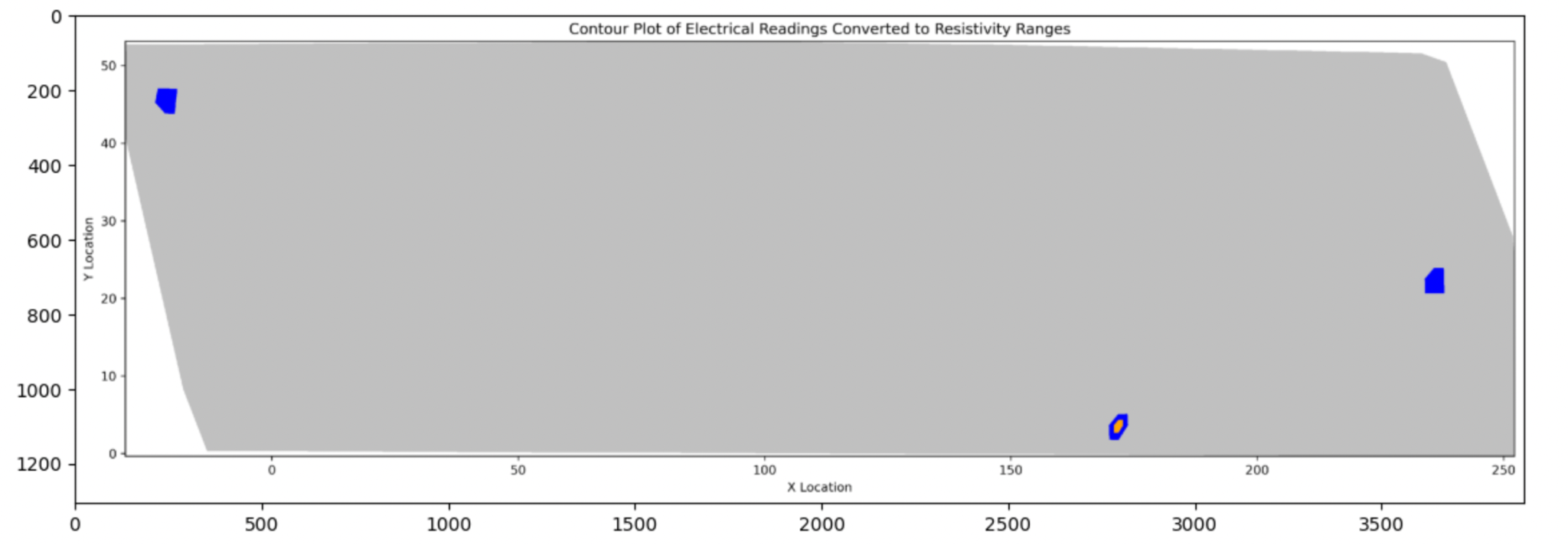
* Highly Corroded: Less than 5 (dark red)
* High Corrosion: Between 5 and 10 (red)
* Moderate Corrosion: Between 10 and 20 (orange)
* Low Corrosion: Between 20 and 40 (blue)
* Very Low Corrosion: Greater than 40 (silver)

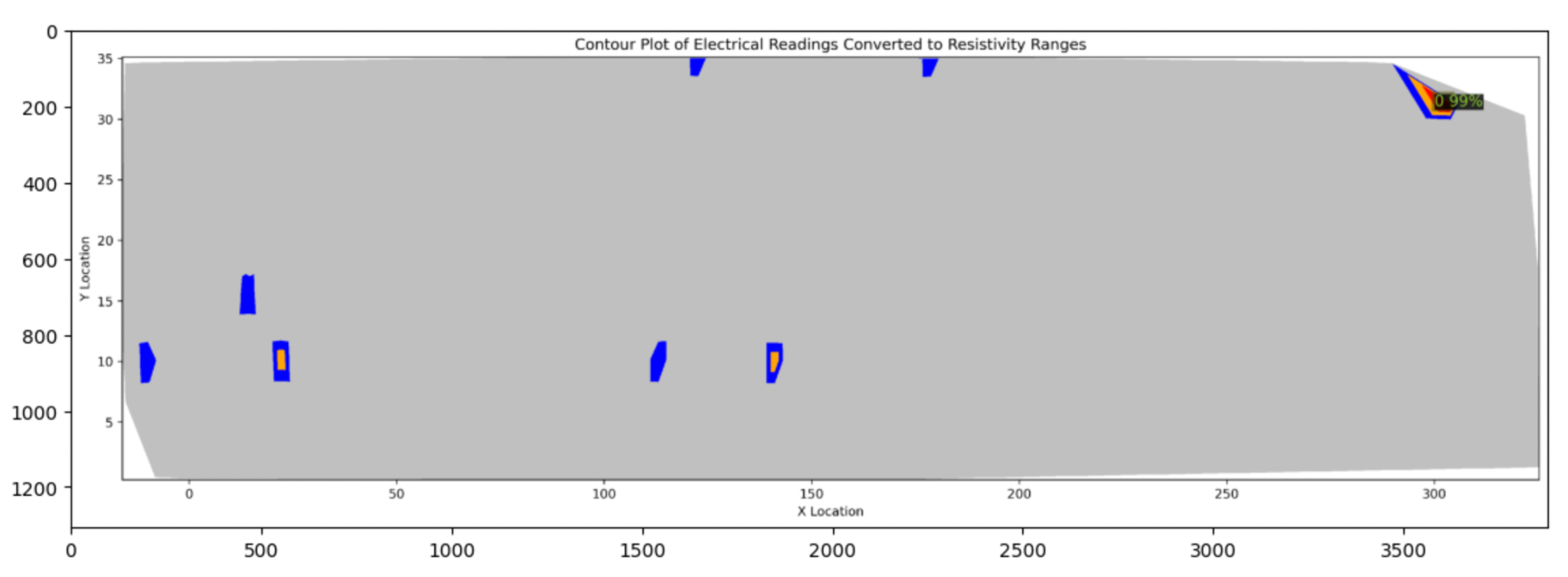
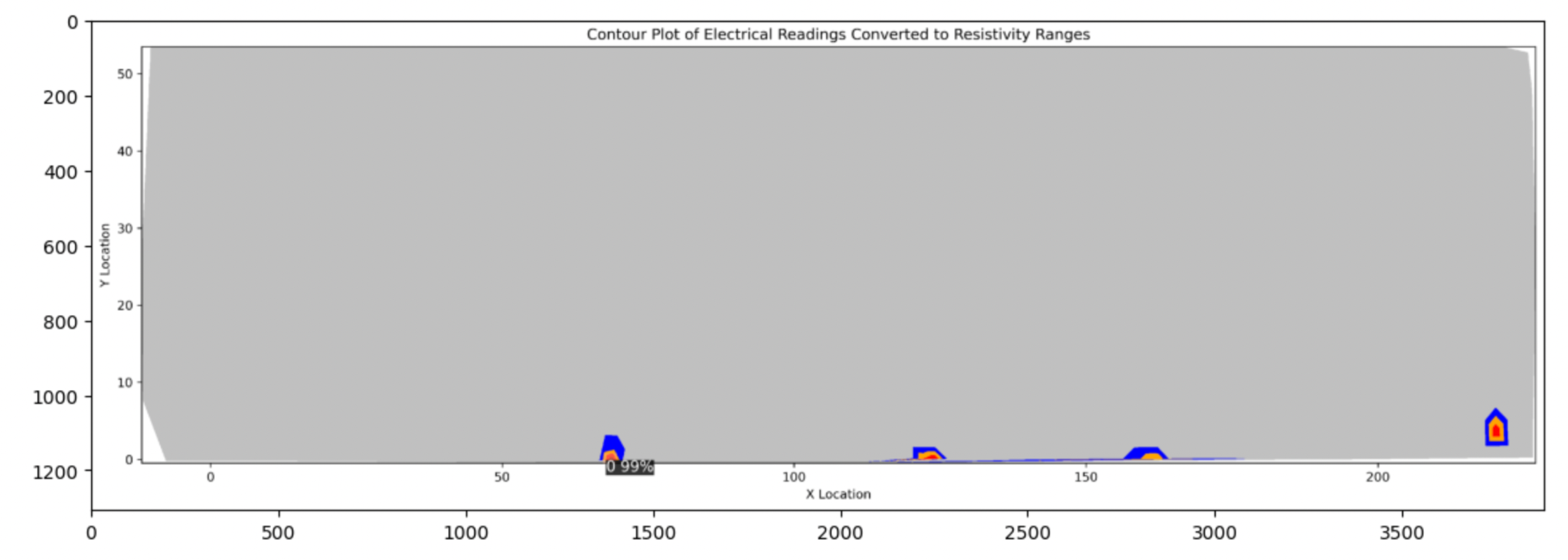
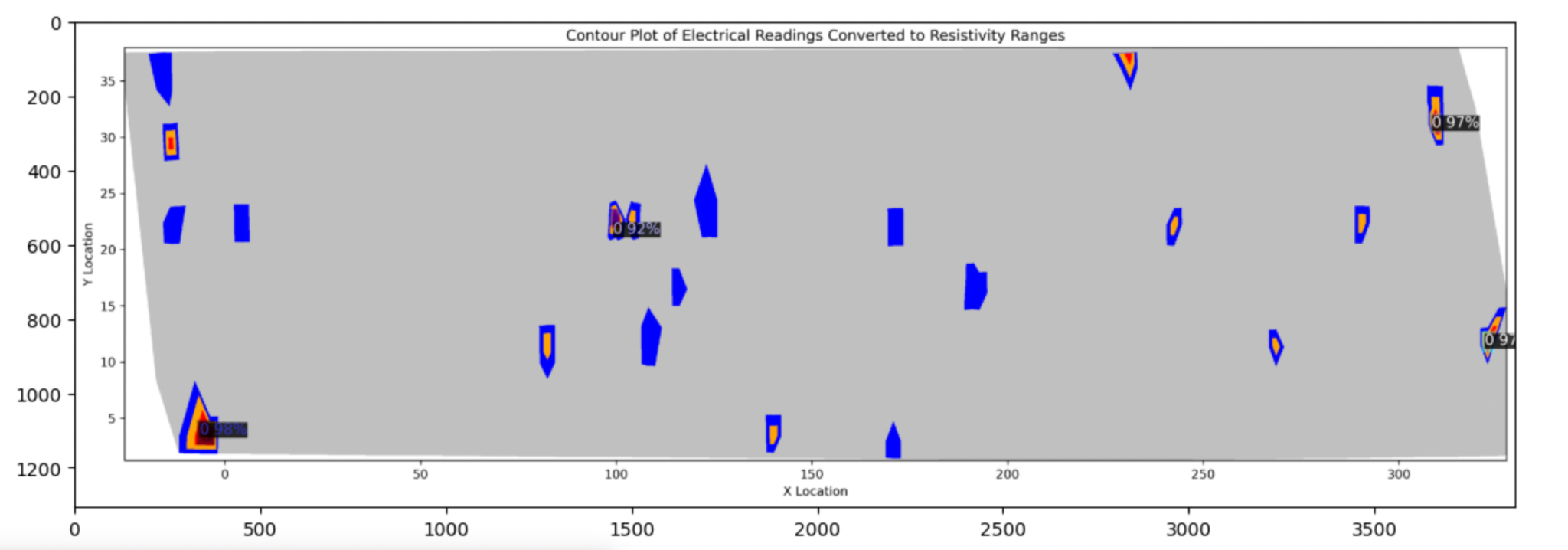
These regions were color-coded accordingly to visually represent the corrosion levels.

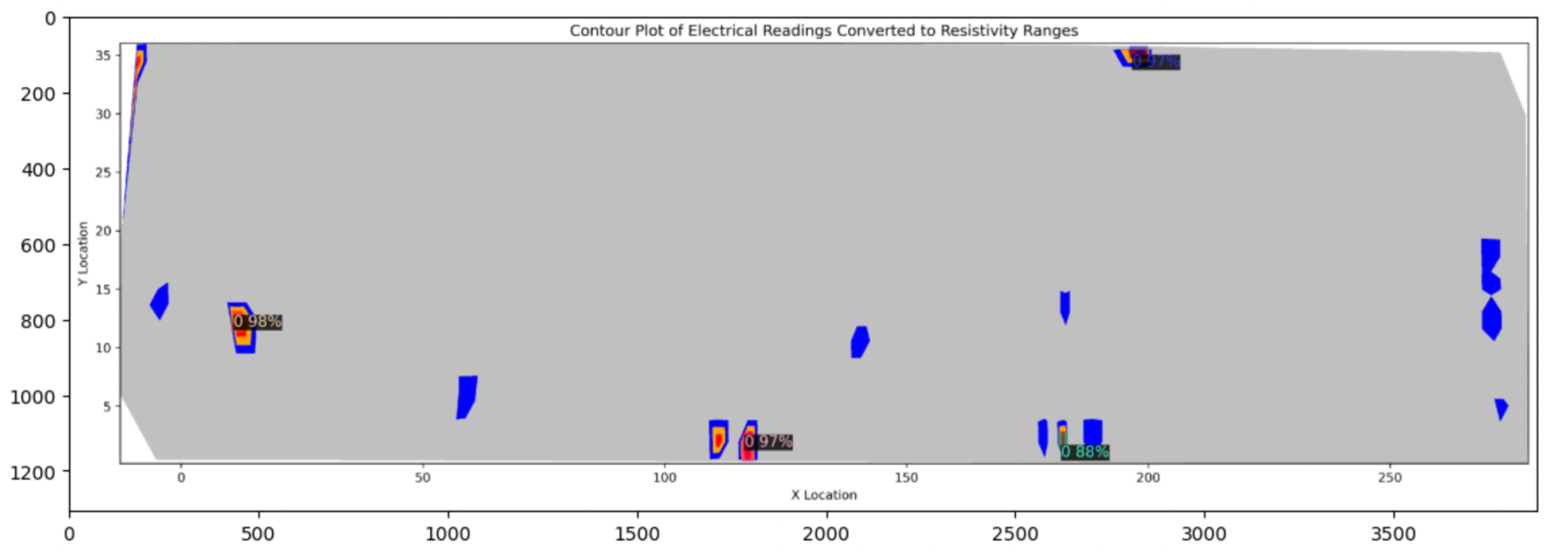
**Model Selection:** To predict the highly corroded areas (dark red regions), the Fast R-CNN model was selected. Bounding boxes were annotated using OpenCV to create annotations.json. The Detectron2 framework was utilized to train the Fast R-CNN model on these annotated images, focusing on accurately predicting the highly corroded areas.

**Model Implementation:** The script installs `pyyaml` version 5.1 and clones the Detectron2 repository from GitHub. It then installs Detectron2 dependencies as specified in the repository's `setup.py`. The annotations from a JSON file are read and organized into a format compatible with Detectron2, including bounding boxes and category IDs. The script registers training and validation datasets with Detectron2, specifying the paths and metadata. The data is trained using a Faster R-CNN model with a ResNet-101 backbone .Training parameters such as batch size, learning rate, and number of iterations are set. The batch size is 2 and 1500 iterations as the data is minimal.

**Results:**After training, the script loads the trained model weights and sets a threshold for detection. It creates a `DefaultPredictor` for inference on the validation dataset. For each image in the validation dataset, predictions are made, and the results are visualized using Detectron2's `Visualizer`. The visualized results, including detected bounding boxes and classifications, are displayed using Matplotlib. The validation images are of Mississippi bridges.

****

****

****

**Challenges:**

The primary challenge was the data, as it was insufficient for training the model. The XML files had varying source names like "ER-Readings," "NDE001Readings," and "NDE001-Readings," which led to incomplete data for some of the 38 bridges. After adjusting the code, we successfully extracted readings for all 38 bridges. The criteria for identifying corroded regions were strict, with a threshold of less than 5. Consequently, there were very few corroded regions across the bridges, and some showed no corrosion. Model selection was also challenging due to computational constraints. After evaluating several models, Faster R-CNN proved to be the most suitable for implementation.