CSCE 5222 Feature Engineering

Project Plan/Report

Group 01

Member: Jack Duffield, Mustafa Syed, Troy Krupinski

Link to Google Colab: [CSCE 5222 Project.ipynb](https://colab.research.google.com/drive/1i1beQesZNdeoWlbyk4uuI6I7F83CQPKE?usp=sharing)

Link to GitHub: [GitHub](https://github.com/TroyKrupinski/CSCE5222_Project)

1. Problem statement In this project, we are developing a Python program designed to automatically identify and localize acceptable components based on visual pattern properties from a sample set of images supplied by a client. The primary goal is to both classify and accurately locate patterns in images.

Components considered "good" have clear stripe or cross patterns, while those without those patterns present are classified as "bad" - sometimes due to overexposure or underexposure obscuring the pattern. By accurately localizing each component, we can ensure that the classification is precise and relevant to specific localized areas within the images.

To refine our methodology and enhance the robustness of our approach, we reference recent work by Chen et al. (2021), who present a comprehensive framework for visual inspection processes, including object detection and localization techniques. The study emphasizes methods like adaptive thresholding and frequency analysis, which are techniques we will be utilizing in this project.

1. Data used

**State the data used in this project including ground truth for evaluation:**

The data we employ in this project is a sequence of 20 grayscale images, each measuring 1000 pixels by 1000 pixels. This set of images is a set of 27 components with each image having different external lighting to simulate real-world scenarios. Each element in the images contained in this dataset has different surface properties that are apparent in the form of stripes, cross patterns, or the lack thereof.

**Expected Ground Truth:** A red x symbol on a white background

Description automatically generated

**Example of Ground Truth Annotation:**

**Image 1:**

**Component 1:**

**Location: Bounding box coordinates (x1, y1, x2, y2)**

**Classification: Good**

**Component 2:**

**Location: Bounding box coordinates (x1, y1, x2, y2)**

**Classification: Bad**

The ground truth for this dataset is established by manually annotating each component based on the visibility and clarity of its surface patterns. Components with distinct stripe or cross patterns are labeled as "good," whereas those without such patterns are labeled as "bad." This manual annotation serves as a benchmark for evaluating the performance of the automated classification system developed in this project.

**“Good” component “Bad” component**



1. Method

Give details of the method and any parameters used.

This project will be implemented using Python and a multitude of libraries; leveraging libraries such as OpenCV for image processing, NumPy for numerical operations, Scikit-learn for machine learning algorithms, and Pandas for data manipulation.

Our methodology involves several key steps: **preprocessing**, **feature extraction**, **component localization**, and **pattern classification using a machine learning classifier (SVM)**. By utilizing SVM, we aim to eliminate threshold-based decision-making and allow the classifier to learn the patterns that distinguish "good" components from "bad" ones.

A high-level diagram of the data flow and system components is provided in **Figure 1**.

1. **Preprocessing**

**Grayscale Conversion:** Convert images to grayscale, if not already grayscale.

**Histogram Equalization:** Application of **histogram equalization** to standardize contrast/lighting across images, which will help mitigate the overexposed and underexposed images in the dataset.

**Noise Reduction:** Application of **image smoothing** to reduce noise which can interfere with pattern detection. To remedy this, we will apply either **Gaussian or median filtering**, which can help us enhance the true patterns of the images in the dataset by removing variance in intensity.

**2. Feature Extraction and Component Localization**

a. Component Localization

* **Object Detection for Localization:**
  + Utilize object detection techniques to locate components within the images. Methods such as Sliding Window with HOG descriptors, Selective Search, or advanced deep learning models like Faster R-CNN can be used for this purpose.
  + The output is bounding boxes around each detected component.

b. Feature Extraction within Localized Components

* **Edge Detection:**
  + Apply edge detection methods like the Canny edge detector within the localized component regions to identify edges.
* **Line Detection:**
  + Use the Probabilistic Hough Line Transform to detect lines within the component regions.
* **Feature Vector Construction:**
  + Line Features: Number of lines, orientations (angles) of lines, lengths of lines.
  + Pattern Features: Presence of intersecting lines (indicative of cross patterns), parallel lines (indicative of stripe patterns).
  + Frequency Analysis: Apply Fourier Transform to capture frequency components indicative of stripe patterns.
* **Eliminating Threshold-Based Decisions:**
  + Instead of using manual thresholds (e.g., number of lines > 5), we extract features and let the SVM classifier learn the optimal decision boundaries.

**3. Pattern Classification Using SVM**

* **Feature Vector Preparation:**
  + Compile the extracted features from each component into a feature vector.
* **Training the Classifier:**
  + Use the labeled dataset (with ground truth annotations) to train the SVM classifier.
  + The classifier learns to distinguish between "good" and "bad" components based on the feature vectors.
* **Classification:**
  + Apply the trained SVM classifier to the feature vectors of the localized components in the test images.
  + The classifier outputs the predicted classification for each component.

**4. Handling Poor Pattern Visibility**

* **Adaptive Methods:**
  + Apply adaptive thresholding and contrast adjustments within the component regions to enhance pattern visibility under poor lighting conditions.
* **Data Augmentation:**
  + Augment the training data with variations in lighting and contrast to improve the classifier's robustness.

(A better illustrated methodology diagram will be in the next update.)

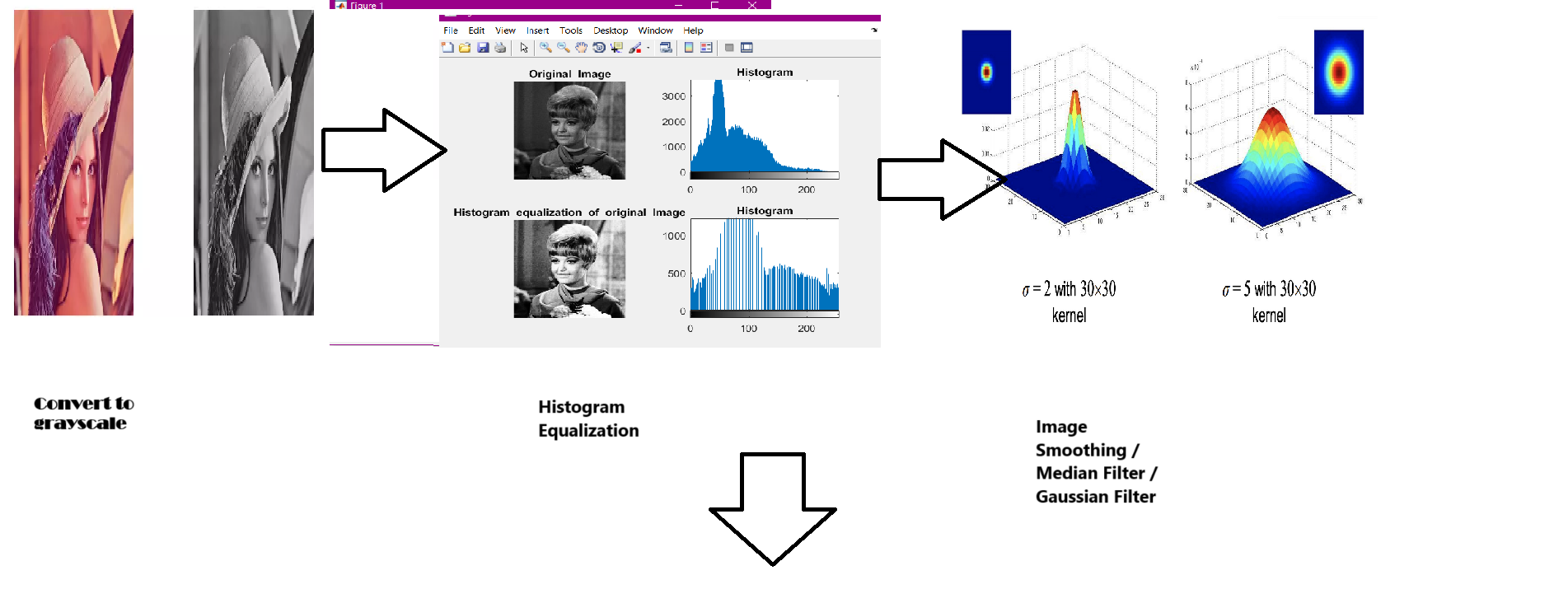
[Input Grayscale Images] --> [Preprocessing] --> [Component Localization] --> [Feature Extraction] --> [Feature Vector Compilation] --> [SVM Classifier] --> [Output Results]

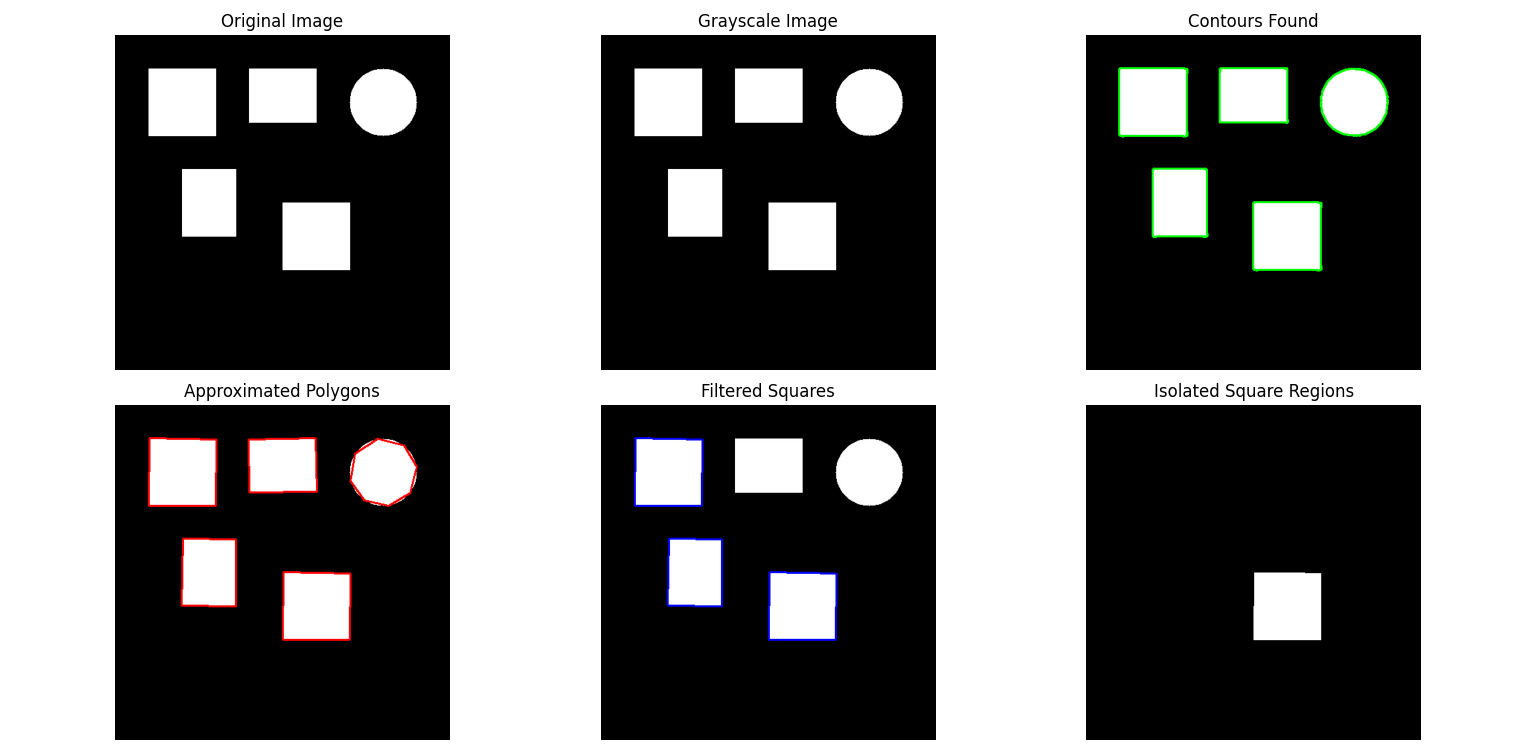
| ^

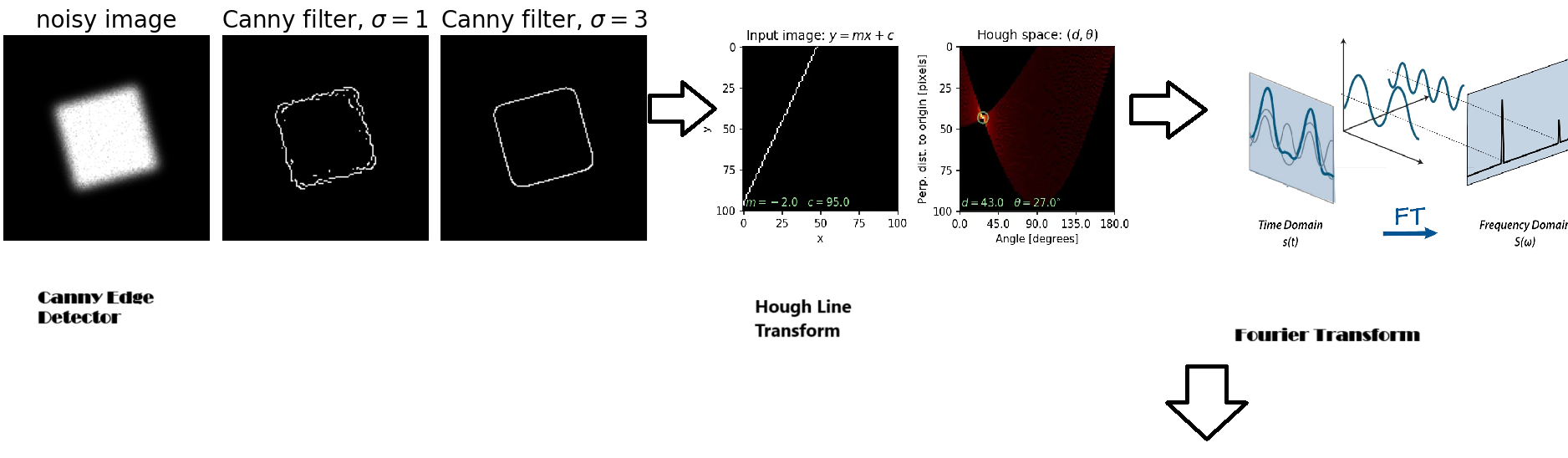
| |

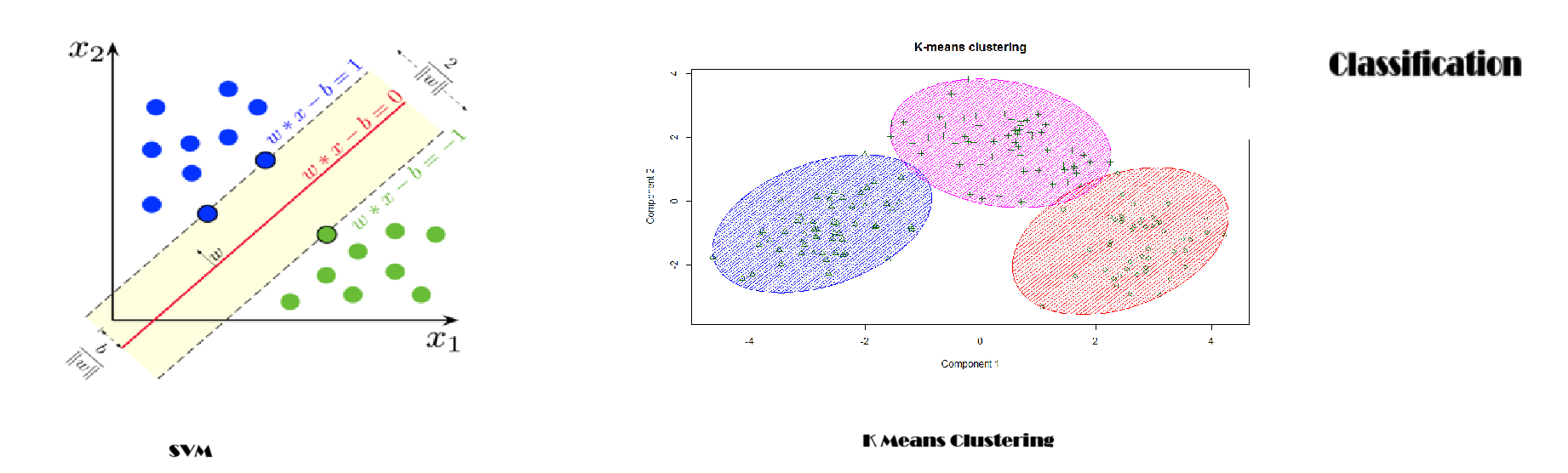
+---------------------------------------- [Adaptive Methods] <-----------------------+

[Evaluation Metrics] <-- [Component Localization] and [SVM Classifier]









**Workflow Summary**

**Preprocess the Image:**

Convert to grayscale.

Apply histogram equalization.

Reduce noise with Gaussian blur.

**Component Localization:**

Use object detection techniques to locate components.

Obtain bounding boxes for each component.

**Feature Extraction within Components:**

Apply edge detection within components.

Detect lines using Hough Transform.

Construct feature vectors without threshold-based decisions.

**Pattern Classification with SVM:**

Train the SVM classifier using extracted features and labels.

Classify components based on the trained model.

Evaluate Performance:

Compare localization and classification results with ground truth **labels.**

Calculate evaluation metrics, including mIoU for localization.

1. Evaluation
2. We will use metrics such as **accuracy** to quantify the proportion of components classified as either “good” or “bad”. **Precision** will also be used to measure the true positives against the false positives. We will also consider scores such as F1 and dues on accuracy & precision.

**2. Performance Metrics for Localization:**

* **Mean Intersection over Union (mIoU)**:
  + **Definition**: mIoU measures the overlap between the predicted bounding boxes and the ground truth bounding boxes, averaged over all components.
  + **Calculation**:
    - For each component, compute the Intersection over Union (IoU) between the predicted bounding box and the ground truth.
  + **Purpose**: mIoU provides a quantitative measure of the localization accuracy of the system.

**3. Validation Techniques:**

* Due to the small dataset, we will employ **cross-validation techniques** to generalize pattern detection under varying lighting conditions.

**4. Error Analysis:**

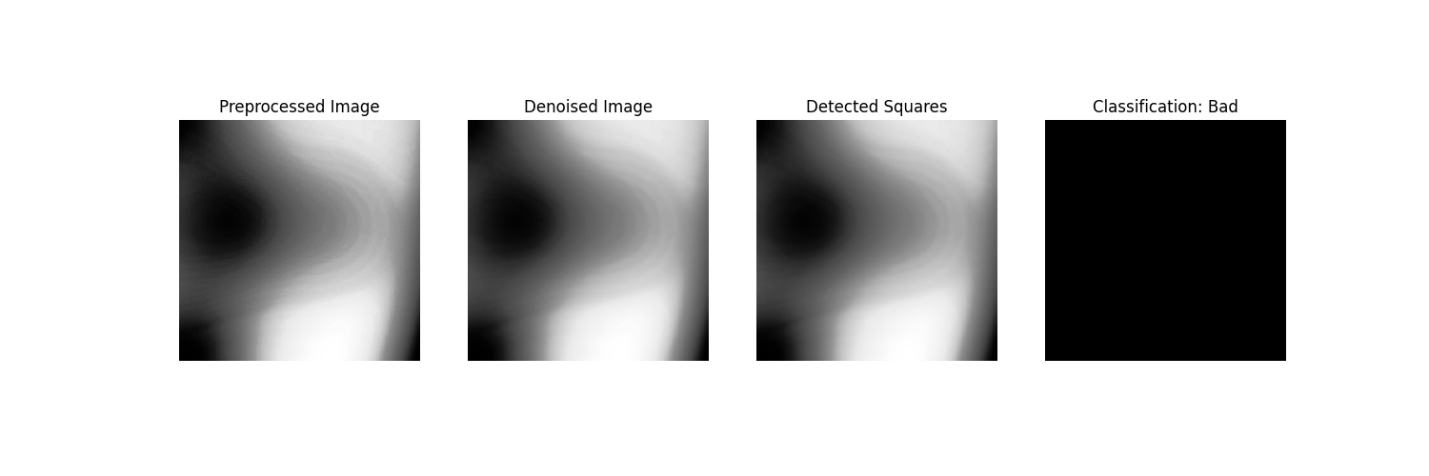
* Examine misclassified components and localization errors to identify shortcomings in preprocessing, feature extraction, or classification stages.
* Adjust parameters and methods based on insights from the error analysis to improve robustness.

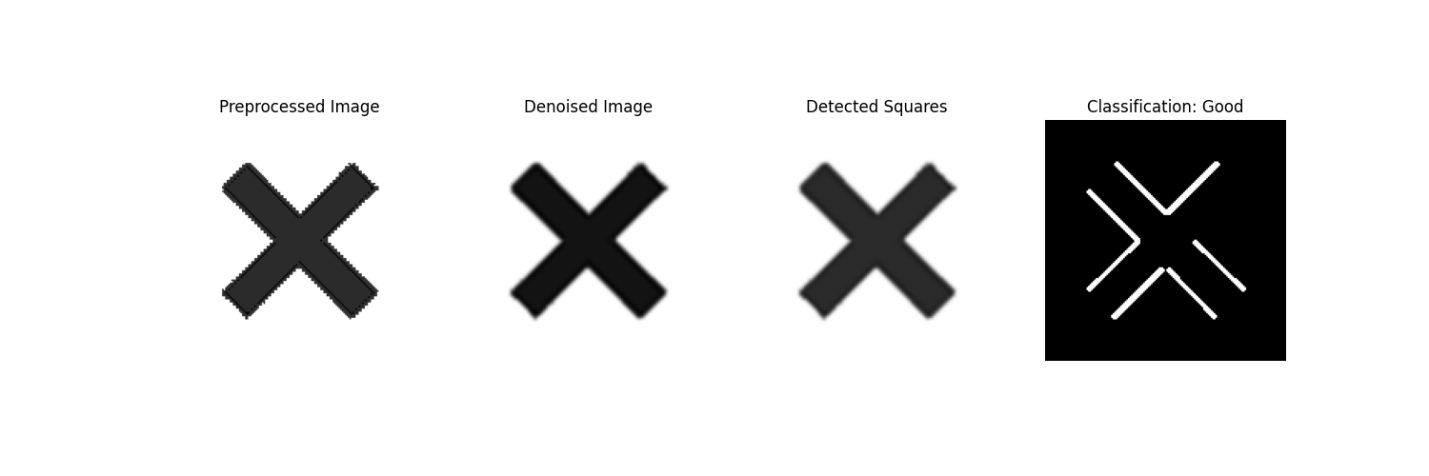
5. Timetable

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activity/Task** | | **Deadline** | | **Group Member(s) Responsible** | |
| Finish /submit project plan | | 11/6 | | Jack, Troy, Mustafa | |
| Preprocessing | | 11/10 | | Mustafa, Jack | |
| Feature extraction | | 11/13 | | Troy | |
| Pattern classification | | 11/13 | | Troy | |
| Evaluation metrics and error analysis | | 11/20 | | Jack | |
| Finish and revise the final report | | 11/20 | | Mustafa, Jack | |
| Prepare presentation | | 12/1 | | Mustafa, Jack | |

**References:**

1. **Chen, H., Zhang, Y., & Zhang, Y. (2021). "Automated Visual Inspection in Manufacturing Using Machine Learning: A Review." *IEEE Transactions on Automation Science and Engineering*, 18(4), 1688-1708.**
2. **Li, X., Chen, Y., & Li, Q. (2020). "Deep Learning-Based Defect Detection for Industrial Applications: A Survey." *IEEE Access*, 8, 157951-157970.**





The following are examples of inserting tables and figures. You can duplicate them and make changes for yours.

**Example output:**

*Unique labels in ground truth: {0, 1}*

*Unique classes in training labels: [0 1]*

*Processing Image: C:\Users\dunke\Desktop\CSCE5222\_Project\fabrizio-coco-u8w5OdRFPn8-unsplash.jpg*

*Image - Ground Truth: 0, Predicted: 1*

*Processing Image: C:\Users\dunke\Desktop\CSCE5222\_Project\good.png*

*Image - Ground Truth: 1, Predicted: 1*

*Unique classes in training labels: [0 1]*

*Processing Image: C:\Users\dunke\Desktop\CSCE5222\_Project\sample\_shapes.jpg*

*Image - Ground Truth: 1, Predicted: 1*

*Processing Image: C:\Users\dunke\Desktop\CSCE5222\_Project\bad.png*

*Image - Ground Truth: 0, Predicted: 0*

*Cross-Validation Classification Metrics:*

*Accuracy: 0.75*

*Precision: 0.67*

*Recall: 1.00*

*F1 Score: 0.80*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Size | Type | Class | Accuracy | Precision |
| Image 1 | 798x482 | .png | Good | 1 | 1 |
| Image 2 | 241x249 | .png | Bad | 0 | 0 |
|  |  |  |  |  |  |