# Zebra Crossing Detection and Classification in Images

## Introduction:

This code implements a process to detect zebra crossings in a set of images using various image processing techniques. The used images are all in the same perspective and taken by a camera mounted on top of a car. Furthermore, it utilizes a deep learning model to classify whether individuals in the images are standing on a zebra crossing or not. The process involves several steps, including image preprocessing, feature extraction, and object detection.

## Image Processing for Zebra Crossing Detection:

In the first part of the code necessary variables are defined and a folder containing a collection of images including zebra crossings is opened. It checks for the existence of the specified folder and extracts a list of image files within it. Key variables like image count, storage containers for different image representations, and flags for detected crossings are initialized.

### Preprocessing Steps:

1. **Resizing and Cropping**: Each image is resized to a consistent size for uniformity and then cropped to focus on only the zebra crossing area.

2. **Grayscale Conversion**: The cropped images are converted into grayscale for a better performance in edge detection.

3. **Edge Detection**: The Canny edge detection algorithm is applied to identify edges within the grayscale images.

4. **Image Thresholding**: A threshold is set to convert the edge-detected image into a binary representation.

5. **Erosion and Morphological Operations**: Morphological operations like erosion are performed to refine the binary images, enhancing the zebra crossing features.

6. Image Overlay: The processed binary images are overlaid onto the original images to highlight the detected zebra crossings.

## Classification Using Deep Learning:

For the classification task, a pre-trained Faster R-CNN model is employed for people detection, followed by a custom-built convolutional neural network (CNN) for binary classification.

### Model Architecture:

1. **Faster R-CNN for Object Detection**: This model identifies people in the images using an ACF-based people detector.

2. **Binary Classification CNN**: A simple CNN architecture comprising convolutional, pooling, and fully connected layers is used for binary classification.

### Training and Classification:

1. **Training Option**: Training options like stochastic gradient descent with momentum (SGDM) and specific hyperparameters are set.

2. **Data Preparation**: An image data store is created with corresponding binary labels indicating whether a person is on a zebra crossing or not.

3. **Model Training**: The binary classifier is trained on the prepared data using the specified CNN architecture and training options.

## Image Processing and Classification Integration:

The processed images are then subjected to the trained models for zebra crossing detection and classification.

### Detection and Annotation:

1. **Processed Image Iteration**: Each processed image is loaded from the designated folder.

2. **People Detection**: The Faster R-CNN model is applied to detect people within the images, providing bounding box coordinates.

3. **Bounding Box Annotation**: Detected people are annotated with green rectangles if standing on a zebra crossing or red rectangles otherwise.

Conclusion:

This code effectively combines image processing techniques with deep learning models to detect zebra crossings in images and classify individuals' positions concerning these crossings. It uses edge detection, morphological operations, and classification models to perform these tasks accurately. The code demonstrates a pipeline for automated zebra crossing detection and person classification in images. While the existing code offers a functional solution, there are potential improvements when looking into the results. There could be made improvements in fine-tuning parameters, considering more sophisticated models, diversifying, and augmenting datasets, and employing comprehensive evaluation methodologies. For example, there is still the problem that the algorithm doesn’t mark all parts of the zebra crossing, just the parts in the very middle and biggest ones. Bringing the mentioned enhancements to reality could lead to an even more accurate and reliable zebra crossing detection and person classification systems.

Summary:

The code showcases a comprehensive workflow that integrates image processing methods for zebra crossing detection and deep learning-based classification to identify individuals' positions relative to these crossings. It involves image preprocessing, edge detection, binary classification, and object detection techniques to achieve the desired results. Overall, it fulfills the given requirements to detect the zebra crossings while deciding if there are people standing on the crossing or not. In future projects we would investigate the discussed weaknesses and potential improvements to make the software even more reliable and precise