CSU44053 Computer Vision | Assignment 1
Brendan McCann | 20332615

# Introduction

The goal of this assignment is to detect pedestrian crossings in images taken from a car's point of view. Firstly, I will describe in depth the theory behind my process, and then present results and analyse both successes and failures.

# 1 Theory of the Processing

## 1.1 Smoothing

To help cope with noise, all my images went through an initial stage of smoothing. In the context of this problem, it was crucial that the edges of the pedestrian crossings remained well-defined and as sharp as possible. The reason for this is because I wanted my edge detection to work as good as possible. As a result, I settled with **median smoothing**, primarily because it does not blur edges too much.

I experimented with multiple iterations of median smoothing, however I found the best results with just one iteration. Additionally, I opted for a filter size of 3x3 as I wanted the edges to be as sharp as possible. With larger filter sizes, I was worried it would corrupt the image too much and deviate too far from the original.

## 1.2 Edge Detection

At first, I experimented with region processing using Mean Shift Segmentation (MSS). However, I found that it took a about two seconds to process the image using MSS. In the context of the problem, I opted for a more efficient approach as it is vital that images are produced rapidly in a car imaging system. This is how I settled with edge processing.

I chose to go with **Canny Edge Detection** as opposed to Boundary Chain Codes, because with Canny I could have more customisation over the results by changing the parameters, whereas with Boundary Chain Codes it was more limiting.

The low and high thresholds were chosen mostly based on trial-and-error. I ensured that my low threshold value was conservative, as this is indicative of the number of contours detected by Canny, and I wanted to be sure that my edge detection would not miss out on any potential pedestrian crossings.

## 1.3 Binary Thresholding

The output of the previous step in the process was an RGB multispectral image, and I wanted a binary image to simplify things further. I used **Otsu Thresholding** to convert to binary, followed by a closing operation. This was done to fill any potential gaps in the edges of pedestrian crossings. From some trial-and-error testing, I found a kernel size of 2 to be optimal for my process.

## 1.4 Connected Components Analysis and Shape Analysis

My next goal was to find contours in the binary image. I used CCA for its simplicity, and also because it showed great results on my initial testing.

After finding all the contours in the image, they would then go through the following filtering process:

*All parameters were chosen through trial-and-error testings.*

1. Filter by Contour perimeter and area
    - Contour Perimeter >= 90
    - 50 <= Contour Area <= 3000
2. Convex Hulls were then drawn around these contours. The Convex Hulls were filtered by area.
    - 500 <= Convex Hull Area <= 3500
3. Convex Hulls were filtered by rectangularity.
    - Rectangularity >= 0.5

After going through this filtering process, we now have a list of filtered convex hulls that potentially identify as pedestrian crossings.

## 1.5 Colour

One distinctive trait of pedestrian crossings are it's white-ish colour. For each filtered convex hull, the mean colour of the convex hull is calculated. This RGB value is compared to the white RGB value by calculating the Euclidean distance between the two 3D vectors. From trial-and-error testing, I found 150.0 to be a forgiving value, but with good results.

## 2 Results and Analysis

## Conclusion