Chapter 3

Conceptual Framework

This chapter elaborates the characteristics of sidewalks for accessibility, an overview of street-level images, and the concepts related to semantic segmentation and object detection that will be used in the study.

3.1 Sidewalk Accessibility

Urban accessibility is defined as the convenience and ease of travel provided to all types of pedestrians, able or disabled, while moving throughout an urban setting. There is an emphasis on the importance of creating accessible infrastructure for disabled persons because a certain street, sidewalk, or path should create an environment that any person can fully participate in. Accessibility may be interpreted differently from country to country depending on the regulations required by local governments, but they all share the same objective of creating equal opportunity for mobility and safety for all pedestrians. As for the sidewalk, Jacobs (1961) mentions that it is a vital transportation infrastructure in cities that allow pedestrians to get from one location to another, but their true role is to keep pedestrians safe, and in order to keep all pedestrians safe the sidewalk should be fully accessible to all pedestrians as well. Based on the ADA, the IRR of the accessibility law of the Philippines, the MMDA regulation 02-28, and Santos (2018) eight principles of sidewalks, accessible sidewalks should be measured by four generalizable metrics, namely *sidewalk design*, *surface texture*, *sidewalk placement*, and *presence of obstructions*.

3.1.1 Sidewalk Design

AYRES (2017) defines accessible sidewalk design as one which enables a PWD to safely cross a street and properly board or exit transportation. The ADA mandates that sidewalks must be easy for a PWD to use, which translates to the convenience it provides for PWD's on a day-to-day setting. Figures 3.1 and 3.2 are examples of accessible design which cater to various types of disabilities. If a sidewalk is designed in a way that it caters to all disabled persons, then it would be considered as an accessible sidewalk not just to PWD's, but to all pedestrians who make use of that sidewalk.The 2010 ADA Standards for Accessible Design details the specific technical requirements for designing visual aids on sidewalks. Designing the sidewalk is a significant process pre-construction to ensure that no renovations in the design of the sidewalk need to be made after it has already be constructed.

Figure 3.1: Colored tiles with elevated domes allow the visually-impaired to recognize road entrances. This was lifted from AYRES (2018).

Figure 3.2: Curb ramps allow people in wheelchairs to cross the street. This was lifted from AYRES (2018).

3.1.2 Surface Texture

According to Santos (2018), the choice of paving material is determined by various environmental and social factors, but regardless of the surface selected the pedestrian zone of the sidewalk must still guarantee full accessibility. This entails maintaining a stable, even, and non-slip surface in both wet and dry conditions. PWD's rely on level surfaces to maintain their balance and guide their movement, therefore there should be no presence of irregularities in the surface which may snag or impede mobility aids such as wheelchairs, canes, or walkers. Surface irregularities may also cause serious accidents for pedestrians. Figure 3.3 highlights an example of good versus bad surface textures.

(a) Pavement is cracked and unsafe

(b) Smooth surface allows wheelchair users to make use of the sidewalk

Figure 3.3: Examples of good and bad sidewalk surface texture. This was lifted from (AYRES, 2017)

3.1.3 Sidewalk Placement

Sidewalk placement is vital in determining overall accessibility and is achieved when a sidewalk is able to create a continuous network for pedestrians, especially those with disabilities, to traverse the city. A continuous network entails that a sidewalk is connected to other paths, creating multiple routes to get from one location to another. Sidewalks should also maintain a significant distance from the road next to it to ensure the safety of its users. When a sidewalk fails to achieve a continuous network of movement, they become unnecessary structures which are seldom used by pedestrians.

3.1.4 Presence of Obstructions

Lastly, we measure the accessibility of a sidewalk by the presence of obstructions found on the sidewalk. Obstructions are classified as any object, structure, or construction project which impedes the movement of an individual trying to pass through a certain sidewalk. The Metropolitan Manila Development Authority (MMDA) (2002) identifies obstructions as a hindrance to the utilization of sidewalks by negatively affecting the smooth flow of traffic. When a sidewalk is heavily obstructed it creates a disjoint in the movement of a pedestrian, sometimes forcing them to make use of the streets themselves which are unsafe and inconvenient. Figure 3.4 shows how construction could be a major obstruction to movement, and this also signifies the importance of sidewalk design as a prerequisite to sidewalk construction. When sidewalks are improperly designed, they require renovation to make them accessible to all users - this creates an obstruction for the period of time it takes to repair a sidewalk and all sidewalks connected to it.

Figure 3.4: Construction projects are obstructions to movement along a path

3.2 Street-Level Imagery

According to Alvarez León and Quinn (2019), street-level imagery is defined as a collection of photographs of roads and roadside assets from the perspective of a vehicle or a pedestrian. It is taken by a 360-degree camera attached to a person or a vehicle such as a car, a motorcycle, or a bicycle. Often, these photographs are processed digitally: stitching the images to form a panorama or a "bubble", georeferenced, and uploaded to a web platform. Several information technology companies partake in collecting street-level imagery of different parts of the world to offer as a service. Major street imagery platform services include Google Street View, Bing Streetside, OpenStreetCam, Look Around, and Mapillary.

In this research, street-level imagery will be collected from Google Street View, which hosts millions of panoramic images collected around the world with over 10 million miles mapped. Aside from its overall completeness, we chose Google Street View for its well-documented API that allowed us to get static street-level images based on coordinates, heading, and pitch. We plan to use static street-level images as our data to train our object detection model, and for users to annotate in the crowd sourcing platform. In the images provided by Google Street View, we aim to detect sidewalk conditions and roadside assets. With this procedure, we accept certain limitations that come with Google Street View images. First, images collected are not real time. Second, locations are limited to those captured by Google Street View and publicly available street-level images.

Figure 3.5: Sample of a street-level image retrieved using the Google Street View API

3.3 Semantic Segmentation

Similar to image classification where objects in an image are assigned classes, semantic segmentation is the process of assigning classes to individual pixels that are found in a single image. Through this process, objects in an image can both be categorized and located through segmentation. Semantic segmentation uses the same class for similar objects which limits it in identifying separate instances of the same class.

Semantic segmentation works by receiving an image and outputting a segmentation map of the image with each pixel having a class label (most of the time the labels are integers). These different class labels are encoded into individual layers

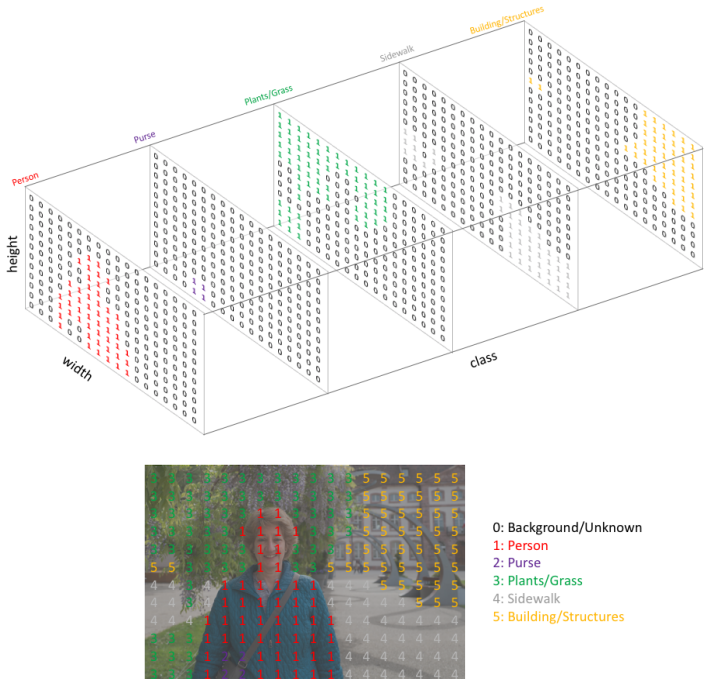

Figure 3.6: Collapsing layers into a segmentation map. This was lifted from Jordan (2018).

and collapsed into one segmentation map.

The general structure of a semantic segmentation architecture implements two types of networks: encoders and decoders. Encoders downsample images which reduces the spatial resolution of the image and the dimensionality of the image resulting in faster processing. Reducing the dimensionality of the image enables a highly efficient discrimination between classes. Sequentially, the decoder upsamples the discriminative features into a full-resolution segmentation map. Upsampling projects the discriminative classes learned by the encoder onto a higher resolution space in the form of pixels. Pooling and unpooling operations are used by downsampling and upsampling respectively. Since downsampling reduces the resolution of an image, pooling summarizes the values of an area in a high-resolution object and projects it into one of the dimensions of the lower-resolution object. On the other hand, unpooling uses a singular value and projects the value to numerous dimensions of the higher resolution object (Jordan, 2018).

A semantic segmentation approach that implements this general architecture is the fully convolutional networks (FCN) introduced in 2014 (Long, Shelhamer, & Darrell, 2015). Fully convolutional networks use upsampling, downsampling and pooling multiple times depending on the variant you are using. FCN-32, FCN-16,

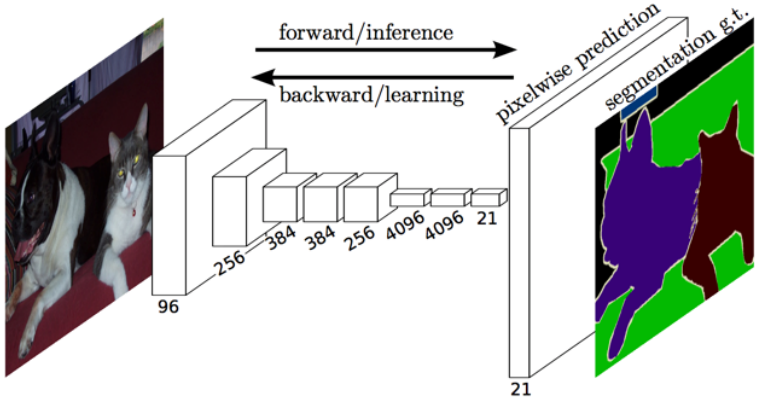

Figure 3.7: Fully convolutional networks (FCN) process. This was lifted from Jordan (2018).

FCN-8 are just some of the popular variants of FCN. The number indicated the amount of upsampling, downsampling, and pooling done on an input. Result of using FCNs are decent but often rough in nature due to the loss of information that occurs from downsampling and upsampling. The introduction of FCN-16 and FCN-8 reduces the loss of information due to the lesser downsampling and upsampling that occurs (Matcha, 2020). Since the introduction of FCN, numerous models have been developed that tweak the architecture of FCN in able to make use of semantic segmentation for their specific use case since the original FCN approach has limitations. An improvement of performance is also another reason why models have begun to tweak and build on top of an FCN architecture.

A metric used in analyzing the performance of a semantic segmentation model is the mean intersection-over-union (IoU). Tiu (2019) states that the IoU is the area of overlap between predicted segmentation and ground truth divided by the area of union between the predicted segmentation and predicted segmentation and ground truth. The metric results in a range of 0-100% where 0 means no overlap while 100% means a perfectly overlapping segmentation.

The performance of FCN-8s on the popular Cityscapes dataset is around 65.3% mean IoU. Newly developed models such as HRNet-OCR, EfficientPS, and DeepLabV3 have a mean IoU of 80-85% (Cordts et al., 2016). These models have continuously tweaked the original FCN model in order to increase their mean IoU. Some of these models With newly developed models having similar performance, we chose to use object contextual representation model, OCR (ResNet-101), as the semantic segmentation model that will be used in our study.

The OCR (ResNet-101) model has a high performance on the Cityscapes benchmark with a mean IoU of 81.08%. Besides this, the OCR(ResNet-101) model is supported by MMSegmentation. MMSegmentation is a PyTorch based open-source semantic segmentation toolbox that will be used in order to perform semantic segmentation. MMSegmentation provides robust documentation and allows for adding new datasets which will be useful when we add our custom dataset. With the use of MMSegmentation, we will be able to implement the OCR(ResNet-101) model when we perform semantic segmentation.

3.4 Object Detection

In defining object detection, we must first define the concept of image classification and object localization. Image classification is the task of assigning a class label to an image. It takes an image as an input, and outputs a class label of the objects present in the image. On the other hand, object localization refers to identifying the location of one or more objects in an image by drawing a bounding box around them. Object detection combines the tasks of both image classification and object localization, classifying and localizing one or more objects in an image (Brownlee, 2019). The output of object detection, as seen in Figure 3.8, is a bounding box around the object being detected, and a class label that describes the object (e.g. person, car, motorcycle).

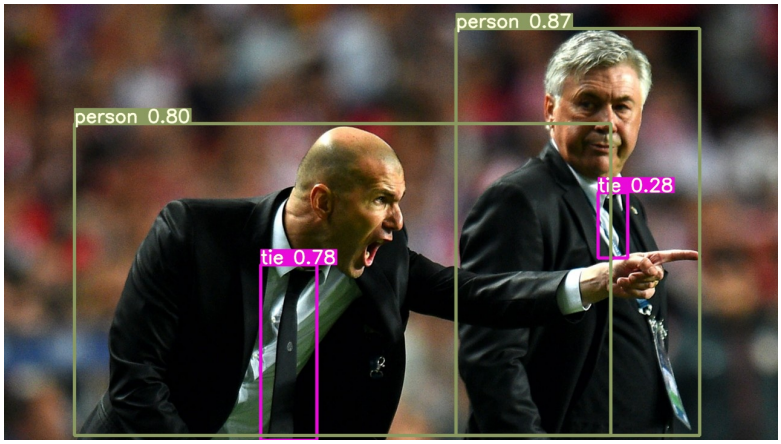

Figure 3.8: Sample output of YOLOv5 detecting individual persons and their ties together with its confidence rating. This was lifted from Jocher (2020).

Object detection is useful in applications of driving assistance and medical imaging. Advanced driver-assistance systems (ADAS) are intelligent systems included in vehicles that use sensors and cameras to detect nearby obstacles and

pedestrians to assist drivers in driving and parking (Kala, 2016). It can provide vital information to reduce vehicular accidents and fatalities caused by human error. Similarly, it is also essential for the technology of self-driving cars or autonomous driving in training the vehicle to detect a known set of objects to know whether it should accelerate, apply brakes, or turn. On the other hand, object detection can be applied in the healthcare industry. It can be used in image diagnosis to identify abnormalities, diseases, and ailments found on the human body.

The most recent and popular computer vision algorithms, also referred to as state-of-the-art models, can be categorized into two main types: one-stage methods and two stage-methods. One-stage methods prioritize inference speed, and example models include YOLO, SSD, and RetinaNet. Two-stage methods prioritize detection accuracy, and example models include Faster R-CNN, Mask R-CNN, and Cascade R-CNN. All models are typically evaluated according to a mean average precision metric, and the most popular benchmark is the Common Objects in Context (COCO) dataset.

The YOLO or You Only Look Once is an object detection algorithm introduced by Redmon, Divvala, Girshick, and Farhadi (2016). It makes use of a single convolutional network to predict bounding boxes and class probabilities from full images in one evaluation. The bounding boxes are weighted by its predicted probabilities. It became popular for achieving high accuracy while also being able to run in real time at a speed of 45 frames per second.

The most recent version of YOLO is the YOLOv5 introduced by Jocher (2020). It is built on a PyTorch framework, compared to the DarkNet framework of YOLOv3 and its earlier versions. Compared to the 45 frames per second of the first YOLO model in 2016, YOLOv5 is able to achieve 140 frames per second. YOLOv5 comes in four different models sizes: YOLOv5s (smallest), YOLOv5m, YOLOv5l, and YOLOv5x (largest). The smallest version is the least accurate among the four, but is considered the "base" model which offers the fastest training time and inference time. The largest version offers the greatest accuracy but takes a longer time to train. The model is trained on the COCO 2017 train/val/test dataset which contains a list of 91 objects (Simalango, 2018). It achieves a score of 55.8 mAP val @0.5 on its YOLOv5s (small) model, and a 67.7 mAPval @0.5 on its YOLOv5x (largest) model.

We decided to use YOLOv5 for object detection as it has a fast training time and inference time while achieving better results than other computer vision models. Comparing it to Faster RCNN model, the YOLOv5s was able to run with a speed of 52.8 FPS compared to the 21.7 FPS of the Faster RCNN. YOLOv5 had a faster inference speed and detected more small objects than the Faster RCNN (Dwivedi, 2020). Additionally, YOLOv5 runs faster compared to its previ-

ous YOLO versions when training it on a custom dataset. According to (Nelson, 2020), testing both YOLO v4 and YOLOv5 on a blood cell count dataset of 364 images across three classes, YOLOv4 took 3.5 hours to train at 1,300 iterations while YOLOv5s took 14.46 minutes to train on 200 epochs. Both models performed similarly with a 0.91 mean average precision @0.50 (IoU threshold set to 50%). Their inference time on single images are also comparable, with the YOLOv4 running inferences in 22ms while the YOLOv5 inferences in 20ms. The author mentioned that although this comparison does not reflect the networks' performance on the COCO dataset (a benchmark for computer vision models), it still shows the performance of both networks on a particular custom task, specifically on the custom dataset used.

With this, we can see that YOLOv5 can train on custom datasets at a fast speed while achieving state-of-the-art results. This will be useful for our study as we aim to train a model with our own custom dataset of street view images to detect objects that affect the accessibility of sidewalks.