Automatic Quantification and Visualization of Street Trees*

Arpit Bahety   Rohit Saluja   Ravi Kiran Sarvadevabhatla   Anbumani Subramanian   C.V. Jawahar
arpitbahety39@gmail.com,rohit.saluja@research.iiit.ac.in,{ravi.kiran,anbumani,jawahar}@iiit.ac.in

Centre For Visual Information Technology (CVIT)
International Institute of Information Technology - Hyderabad (IIIT-H)
Gachibowli, Hyderabad 500032, INDIA

ABSTRACT

Assessing the number of street trees is essential for evaluating urban greenery and can help municipalities employ solutions to identify tree-starved streets. It can also help identify roads with different levels of deforestation and afforestation over time. Yet, there has been little work in the area of street trees quantification. This work first explains a data collection setup carefully designed for counting roadside trees. We then describe a unique annotation procedure aimed at robustly detecting and quantifying trees. We work on a dataset of around 1300 Indian road scenes annotated with over 2500 street trees. We additionally use the five held-out videos covering 25 km of roads for counting trees. We finally propose a street tree detection, counting, and visualization framework using current object detectors and a novel yet simple counting algorithm owing to the thoughtful collection setup. We find that the high-level visualizations based on the density of trees on the routes and Kernel Density Ranking (KDR) provide a quick, accurate, and inexpensive way to recognize tree-starved streets. We obtain a tree detection mAP of 83.74% on the test images, which is a 2.73% improvement over our baseline. We propose Tree Count Density Classification Accuracy (TCDCA) as an evaluation metric to measure tree density. We obtain TCDCA of 96.77% on the test videos, with a remarkable improvement of 22.58% over baseline, and demonstrate that our counting module's performance is close to human level. Source code: https://github.com/iHubData-Mobility/public-tree-counting.

CCS CONCEPTS

• **Computing methodologies → Computer vision**.

KEYWORDS

counting street trees, tree detection, kernel density ranking

ACM Reference Format:
Arpit Bahety   Rohit Saluja   Ravi Kiran Sarvadevabhatla   Anbumani Subramanian   C.V. Jawahar. 2021. Automatic Quantification and Visualization of Street Trees. In *Proceedings of 12th Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP'21),* Chetan Arora, Parag Chaudhuri, and Subhransu Maji (Eds.). ACM, New York, NY, USA, Article 23, 9 pages. https://doi.org/https://doi.org/10.1145/3490035.3490280

*Produces the permission block, and copyright information

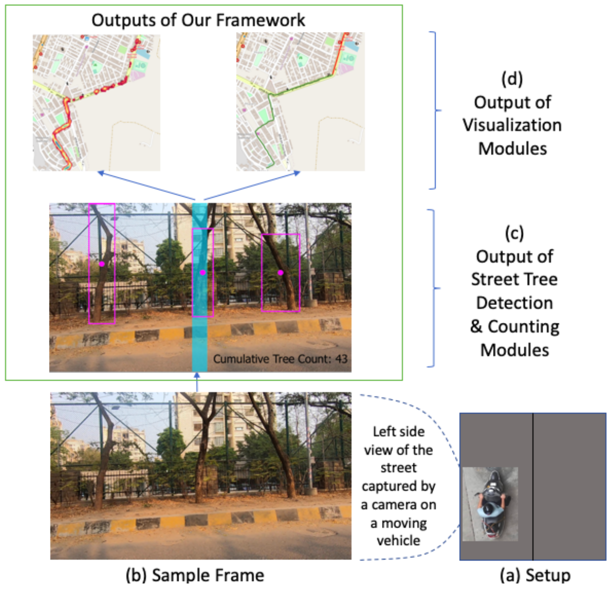
Outputs of Our Framework
(d) Output of Visualization Modules
(c) Output of Street Tree Detection & Counting Modules
Cumulative Tree Count: 43
Left side view of the street captured by a camera on a moving vehicle
(b) Sample Frame
(a) Setup

Figure 1: Output of different modules of our framework for data collection, quantification, and visualization of street trees. Such a framework with city-wide visualizations can help in urban planning. (a) shows data collection platform (explained in Section 3), (b) sample of the captured frame, (c) output from detection and counting modules where detections are shown in pink and counting range by the blue bar, and (d) illustrates the final visualizations based on two different measures of tree density on the routes.

1 INTRODUCTION

Increased urbanization is destroying natural ecosystems and degrading the environment of urban areas. The services provided by urban trees can mitigate some of these problems [19, 26]. An essential part of urban trees is street trees, which provide numerous services. Studies in tropical cities have found that street trees help reducing air temperature, humidity, and air pollution [23]. For example, in Bangalore, India, an experimental study showed that afternoon ambient air temperatures were 5.6 °C lower in roads lined with trees, and road surface temperatures 27.5 °C lower than those measured in comparable tree-less streets [23]. Street trees also reduce flooding and stormwater damage [7, 22, 24], attenuate traffic noise [10], and increase perceived neighborhood safety [14]. The above indicates that having a system to assess street trees is important. Civic or municipal authorities could use such a system to identify tree-starved streets and employ afforestation efforts. Over the years, various works have been developed to quantify urban greenery, detect and catalog trees. However, many of these works rely on aerial or satellite imagery, which is not the fittest setting for assessing street trees [28], or they do not provide high-level visualizations, which are imperative for authorities to understand tree coverage and take adequate measures. In this paper, we propose a system to detect and count street trees and showcase the number of street trees in a meaningful way. We also explain our data collection setup and annotation guideline for street trees (refer Section 3). We present two different approaches to visualize the results, which are explained in Sections 4.2 and 4.3. These visualizations can help achieve a high-level view of the street tree distribution in cities, towns, or villages. The outputs from different modules of the entire framework are briefly explained in Figure 1 (refer to Figure 4 for further details). The core of our system is modules for the detection and counting of street trees. By using our unique street tree annotations and tweaking the YOLOv5 model, we obtain an mAP of 83.74%. We then develop a novel counting algorithm to quantify the street trees. We evaluate the street tree counting using a metric defined as tree count density classification accuracy (refer Section 5) where we obtain an accuracy of 96.77% which is a 22.58% improvement over the baseline.

The key contributions of this work are:

- A setup carefully designed to count roadside trees and a unique tree annotation strategy, which also helps to solve the intra-class occlusion problem [27]. Unlike other works on tree detections, our annotation process is designed to avoid ambiguities in the tree boundaries (refer Section 3). Hence, we achieve highly reliable detections and counting estimates.
- A framework to quantify street trees based on object detection techniques. We also develop an algorithm to quantify the street trees.
- High-level visualizations based on the density of trees on the routes and Kernel Density Ranking (KDR).

2 RELATED WORK

We demonstrate related work in three categories. The first is the canopy cover estimation. The second is work related to tree detection, where the focus is only predicting boxes around trees. The third is tree assessment, where along with detecting trees, the trees are quantified, and results are showcased in a meaningful way to assess the trees.

Canopy Cover Estimation. Canopy cover estimation is a way to measure the greenery in cities. It is different from quantifying individual trees in a street. Existing methods that calculate canopy cover primarily rely on the original colour thresholding and clustering method to filter for possibly misidentified green specks [5, 13]. However, these methods tend to predict green objects with no vertical vegetation as green cover (false-positives). Moreover, they cannot predict non-green parts of vertical vegetation such as branches and yellow leaves as green cover (false-negatives). One of the most recent and popular works in canopy cover estimation that avoids the previously mentioned problem is by Cai et al. [3]. They propose a semantic segmentation (PSPNet [31]) and a direct end-to-end deep convolutional neural network to estimate Green View Index (GVI) to quantify urban canopy cover.

Tree Detection. Earlier works on tree detection use Light Detection And Ranging (LiDAR) [4, 12, 17] or a combination of LiDAR and aerial imagery to detect trees [15] and are mostly focused on delineating trees in forests. However, using LiDAR is costly, and obtaining such aerial images requires expensive flights or drone campaigns. One of the first works to use only camera based images for tree detection is by Yang et al. [29]. In this work, the authors first classify aerial RGB images using a pixel-level classifier to a {tree, non-tree} label based on a set of visual features. In the second stage, they locate individual trees and provide an estimate of its crown size. However, this detection approach falls prey to locally indistinguishable objects such as grass and river. It remains unclear whether it will scale to entire cities with strongly varying tree shapes. Moreover, all the works mentioned hitherto have aerial views and not street views. Data in street view is important for assessing street trees because overhead imagery cannot represent the street-level and resident perspectives of street trees [28]. Two relevant works that performed tree detection on street view images are by Itakura et al. and Xie et al. [8, 27]. Itakura et al. [8] performed automatic tree detection on street-view images from three-dimensional images reconstructed from 360-degree spherical camera using YOLOv2. Xie et al. [27] are the closest to our work in terms of tree detection. The authors present a novel framework for tree detection, leveraging state-of-the-art deep learning based detection methods, along with two innovations in training loss definition and network module designation to handle the problem of inter-class and intra-class occlusion. They obtain their data from panoramic images captured from 22 roads in the city of Nanjing in China using five fisheye lens. They take only two images on the left and right side from the panorama. However, in these works, the annotation process involves drawing the bounding box over a complete tree image, which carries complications (as we explain in Section 3). In contrast, we annotate only the trunks for detection. Section 3 describes our method and its advantages.

Street Tree Assessment. Only detecting street trees would not provide any real-world application and thus, assessing street trees and showcasing the results in a meaningful way is important. One work that quantifies street trees is by Yao et al. [30]. In this work, the authors aim to detect individual trees from satellite data and then perform tree counting based on density regression. They test four different Deep Neural Network (DNN) models on a tree counting dataset constructed with remote sensing images of 0.8m spatial resolution in distinct regions. However, satellite imagery is not conducive to assessing street trees, as mentioned previously. A different work by Branson et al. [2] performs tree detection and cataloging using street view images. In this work, the authors propose an automated, image-based system to build up-to-date tree inventories at a large scale, using publicly available aerial images and panoramas at street-level. Two important things to note are — First, they use multi-view detection. Secondly, the target outputs and training annotations in their object detection model (Faster RCNN) are geographic coordinates (latitude/longitude) rather than bounding boxes. They develop a technique to interchange between coordinates and bounding boxes. This work is closest to our work because they are also detecting individual trees from a street view. However, this work only pertains to the detection of street trees and

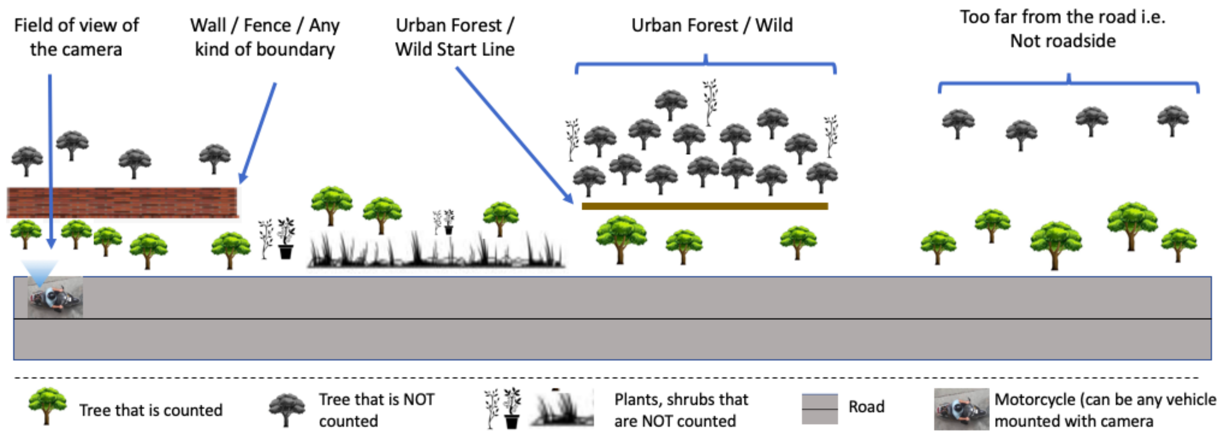
Field of view of the camera
Wall / Fence / Any kind of boundary
Urban Forest / Wild Start Line
Urban Forest / Wild
Too far from the road i.e. Not roadside
Tree that is counted
Tree that is NOT counted
Plants, shrubs that are NOT counted
Road
Motorcycle (can be any vehicle mounted with camera

Figure 2: Depiction of trees that are counted and not counted by our system.

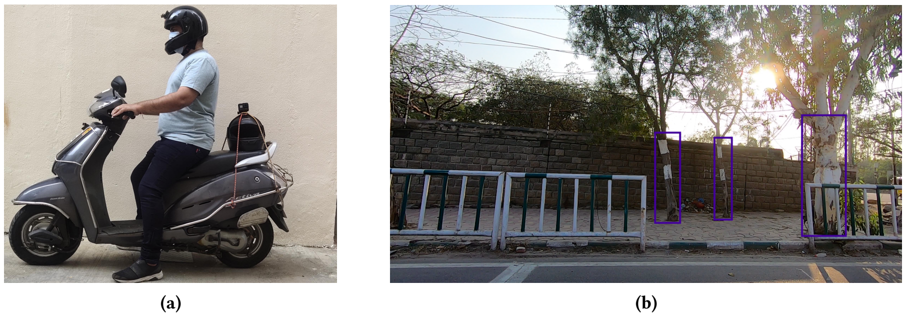

Figure 3: (a) Our setup: we fix the camera over the top of a helmet mounted on the motorcycle's back seat, facing the sidewalk (when on the road). (b) Sample image, annotated with blue boxes, obtained from the camera.

building up the inventory. There is no tree counting and depiction of which areas have fewer trees and which have adequate. Moreover, they use publicly available street views that are not available for many places like India. Furthermore, they detect trees and build their inventory using static images. We take advantage of a vehicle driving and capturing the side view of the street and such a setup helps to count the trees efficiently.

To the best of our knowledge, ours is the first work that performs street tree quantification through street-view images and provides high-level visualizations of the number of trees in various city streets. Such quantification and visualizations can potentially help in urban planning and afforestation efforts.

3 SETUP AND DATASET

Setup. Urban areas have trees in different types of regions, for example, roadside, urban forests or even parks. Many places in India have urban forests starting right next to a road. Thus, one of the challenges that we face is defining which trees would be assessed by our system. After a thorough analysis, we have developed a scheme that delineates the kind of trees counted and not counted by our system (Figure 2). As shown in this figure, we only count the trees that are on the roadside, and avoid counting plants, shrubs, trees in urban forests and trees behind walls or private properties. The data collection setup is an important part of our work. Our focus for the setup is to have low cost and good accuracy for detection and counting. The setup of the camera and an example image from the camera with annotation is shown in Figure 3. Some key points of our setup are -

(1) We mount a camera on a moving vehicle. The camera is facing towards the sidewalk[1]. Refer Figure 2 for the field of view of the camera. Since the camera moves in a uniform direction and the object of interest (tree) is static, this setup is ideal for the counting algorithm we explain in Section 4.1.
(2) Through a unique way of annotating trees, we tackle the problem of intra-class occlusion [27], where canopies of trees occlude each-other.
(3) By having the vehicle move on the lane closest to the sidewalk, we avoid inter-class occlusion (by other objects like vehicles, persons) considerably [27].
(4) Public vehicles and surveillance vehicles can be used to ensure negligible additional costs. However, these vehicles may not always stay on the lane closest to the sidewalk, leading to a trade-off between cost and accuracy.

Thus, our unique setup helps the tree detection model and the counting algorithm to work well. As we will see in Section 5, we achieve accuracy close to human-level on a proposed metric.

Dataset. Existing object detection datasets with trees as one of the classes include OpenImages dataset [11]. The OpenImages

[1]In countries with left-hand traffic (like India), the camera will be facing towards the left side, and in countries with right-hand traffic (like the USA), the camera will be facing towards the right side.

Table 1: Detailed information about the tree detection and counting dataset. The test images are to evaluate the detection results. The five test videos, covering approximately 25 km of roads, are to assess the counting results. The dataset is collected from streets in Hyderabad and Delhi. The resolution of all the images and videos is 1920×1080.

Group	# Images/Videos	# Trees
Training Images	1110	2265
Test Images	185	302
Test Videos	5	796

dataset contains the bounding boxes annotations around the complete tree (including the trunk and the canopy). Annotating an entire tree creates a problem because the canopy is often not distinguishable, leading to ambiguity during annotation. However, the trunks are easily separable and thus provide a more straightforward and more consistent annotation task. An example of an instance where the canopies of multiple trees are not easily separable can be seen in Figure 3 (b). Here, it is difficult to mark the horizontal extents of the canopies of the three trees.

Moreover, our work focuses on street tree quantification rather than just street tree detection. The extent of each tree is inconsequential for the problem of street tree counting. What is important is to count every tree and obtain high tree detection and counting accuracy, and for this, annotating just the trunk is favourable. Furthermore, annotating just the trunk also helps resolve the problem of intra-class occlusion because, unlike tree canopies, tree trunks are usually separable. Lastly, we also believe that detecting trees using only the trunks would help scale up to different species of trees better as the canopies of different species vary a lot, but their trunks are visually similar. However, we have not conducted any experiments on this as it requires additional annotation of tree classes by experts, and hence would be part of our future work. In this work, we focus on counting the trees, irrespective of their types.

Due to the above reasons, we develop a new dataset for street trees with the bounding boxes around the trunk. To collect the data, we first captured videos from a GoPro mounted on a motorcycle and driven on various roads of Hyderabad and Delhi. The video is captured at 30 fps and at a resolution of $1080p$. To create the training dataset for the tree detection model, we then extract images from the videos at 1 fps. There are 1295 images with 1110 images in the training set and 185 images in the test set. To reduce the number of false positives (like poles, bus stand and pillars), we have carefully included 12% of the entire training data as background images. We also have five test videos to evaluate counting results. Details about the dataset are given in Table 1.

Correct and consistent annotations are imperative for deep learning models. Since the tree trunks have inconsistent shapes, we develop certain annotation rules to have consistent bounding boxes. The top edge of the bounding box is made where the trunk starts branching out, or the trunk stops being visible, or where the leaves start appearing. For partially occluded trees, we only mark the visible part of the trunk. If other objects (humans, poles and so on) bisects the trunk and occupies less than $15 - 20\%$ of the bounding box area, we include that in the bounding box. We exclude trees if less than $10 - 20\%$ of the object is visible, such that we cannot be sure if it is a tree. These images are annotated by a professional annotation lab using the Computer Vision Annotation Tool (CVAT) [20]. An example of our tree annotation is shown in Figure 3 (b).

4 THE SYSTEM

The working of our system is as follows - we first pre-process the input video and then input it to a detection module. The captured videos could be long with a single video covering as much as 25 km and covering multiple streets. Since the end goal of our system is to provide meaningful interpretation in the form of maps of the tree density for individual streets, it is desirable to segment the videos into smaller distances. Thus, before the video is input into the detection module, we divide the input video into smaller segments. In the detection module, our object detection model outputs bounding boxes for detected trees. These bounding boxes along with a .gpx file are input to a tree counting and GPS extraction module. Our counting algorithm gives the cumulative count of trees at each frame and the final count at the end of the video. Moreover, whenever a tree is counted, the Global Positioning System (GPS) coordinates of the vehicle at that point are stored. Thus at the end of processing the video, we have fairly accurate GPS coordinates of the street trees. Now, to showcase the number of street trees in a meaningful way, we generate two different kinds of visualizations. First, the category map takes the following as input - tree count of the route, GPS coordinates of the route and our predefined scale (tree count density index, refer Section 4.2) and outputs a map where the routes are coloured according to the index. Second, the density map takes the following as input - the GPS coordinates of the counted street trees and uses the Kernel Density Ranking (KDR) algorithm [6] to generate a map that showcases the density of street trees in a route. Note the detection and counting modules provide a separate output video with roadside tree detections and counts in mp4 format. The entire pipeline is showcased in Figure 4.

4.1 Detection and Counting

In the detection module, the model tries to predict bounding boxes for all instances of trees in each frame. It is important to note that we have only one class - tree. We train a YOLOv5 model [9] on the dataset described in the previous section. The task of tree detection is simplified due to the way we annotate trees. Since YOLOv5 is trained to detect only the trunks of trees and the trunks are usually separable, the detection model does not get confused in detecting individual trees where the canopies of those trees are not separable. Moreover, road scenes have other objects such as poles and brown coloured fences. that can be mistaken for a trunk and lead to false positive detections. To avoid this, we incorporate many such objects in our training dataset, and thus, the YOLOv5 model learns to distinguish trunks from other similar-looking objects.

There are various reasons for choosing YOLOv5 for tree detection. Firstly, YOLOv5 incorporates Cross Stage Partial Network (CSPNet) [25] into its backbone and in the neck. CSPNet helps to achieve a richer gradient combination while reducing the amount of computation, which ensures the inference speed and accuracy are high and reduces the model size. In the tree detection task, fast detection speed is imperative if the video capturing vehicle moves

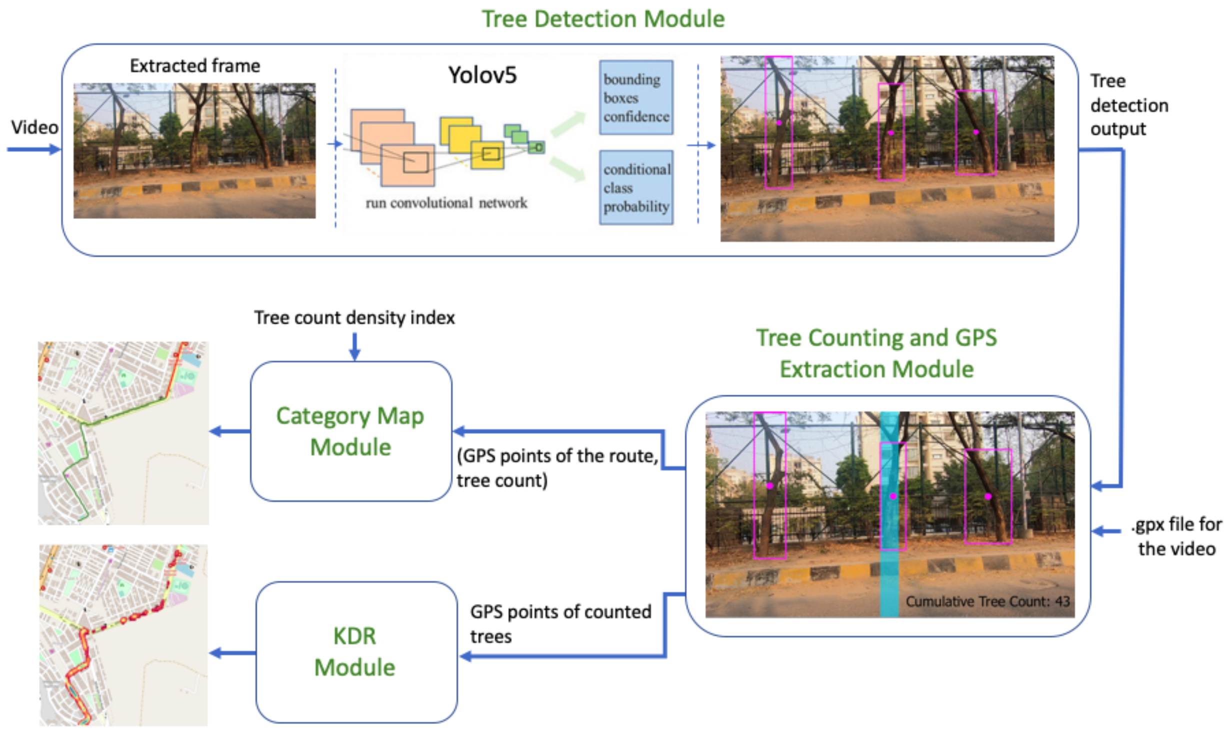
Tree Detection Module
Extracted frame
Video
Yolov5
run convolutional network
bounding boxes confidence
conditional class probability
Tree detection output
Tree count density index
Tree Counting and GPS Extraction Module
Category Map Module
(GPS points of the route, tree count)
.gpx file for the video
Cumulative Tree Count: 43
GPS points of counted trees
KDR Module

Figure 4: The pipeline of our system. The tree detection module detects trees in each frame of a video. In the tree counting and GPS extracting module, the detected trees are counted and their GPS locations stored. The blue box is the counting range (explained further in Section 4.1). The cumulative tree count is updated in real-time in the video, as shown at the bottom right. Finally, the two visualization maps are shown at the bottom left.

at high speeds. Moreover, to obtain accurate tree density maps, maintaining high accuracy is also essential. Furthermore, we also envision that this system will be deployed on resource-poor mobile devices. In such cases, compact models are essential. Secondly, the head of YOLOv5 generates 3 different sizes (18×18, 36×36, 72×72) of feature maps to achieve multi-scale [16] prediction, enabling the model to handle small, medium, and big objects. Street trees usually have trunks of varying sizes in terms of width as well as height. Multi-scale detection ensures that the model can detect all such trees. YOLOv5 also auto-learns custom anchor boxes such that the anchors are particularly adapted to our tree dataset and this helps to improve the detection results. It also performs various augmentations such as mosaic augmentation during training which greatly helps to generalize. Moreover, we experiment with other models like - YOLOv4 [1] and Faster-RCNN [18] and we obtain the best results with YOLOv5 (refer Section 5) while it also takes the least training time.

The predictions from the tree detection module are input to the tree counting and GPS extraction module. A tree is added to the count whenever the centre of the bounding box of a tree falls within a counting range. We define a counting range as a rectangular box with width as a fixed percentage of the image size and height equal to that of the image (refer filled blue rectangle at the bottom-right image in Figure 4). The width of the counting range needs to be carefully decided. If the width is too low, system may miss counting a detected tree since its bounding box centre would never fall in the counting range. If the counting range is too broad, then the same tree would be counted multiple times which we term as double counting. It is easier to handle the problem of double count than missed counts. Thus, we ensure that the width of the counting range is large enough such that the center of the bounding box of a tree detection falls within it at least once in all the consecutive frames containing the detections of the tree (being counted). As a result, we will face the double count issue, which is solved using a simple technique that is explained next. We compare a newly detected bounding box in the counting range to certain previously counted bounding boxes. Suppose the new bounding box has a similar box size as any of the previously counted bounding boxes (within a predefined time period). In that case, we consider that it is the same object as one of the previously counted objects and has already been counted, so we ignore it. The bottom right image in Figure 4 illustrates the counting mechanism. The complete counting algorithm is explained in Algorithm 1. Furthermore, the counting and GPS extraction module takes a .gpx file corresponding to the input video. The GPS location of the entire route can be extracted from this file. In this module, we store the GPS locations of every tree at the frame in which it gets counted.

4.2 Count Density Index and Category Map

In this section, we explain how we develop the tree count density index and its interpretation. We also explain the first visualization map - category map (refer bottom-left image in Figure 4). From

```
Let:
$box$ be a bounding box defined by <centre, height, width>
$bbs$ be list with each item as $box$, it contain all bounding
  boxes in a frame
$dict$ be a dictionary <box, frames_counter>
$tree\_counter \leftarrow 0$
$iou\_th \leftarrow 0.5$
$next\_frames\_to\_consider \leftarrow 7$
**Input** : $\langle Video \rangle$
**for** $frame$ in $Video$ **do**
    **if** $dict$ is not empty **then**
        **for** $key, value$ in $dict$ **do**
            $dict[key]-$
            **if** $dict[key] == 0$ **then**
                $dict$.pop($key$)
            **end**
        **end**
    **end**
    **for** $bb$ in $bbs$ **do**
        $new\_obj \leftarrow$ True
        **if** $bb$.centre lies in counting_range **then**
            **if** $dict$ is empty **then**
                $dict[bb] \leftarrow next\_frames\_to\_consider$
                $tree\_counter++$
            **end**
            **else**
                **for** $key, value$ in $dict$ **do**
                    **if** $IOU(bb, key) \geq iou\_th$ **then**
                        $new\_obj \leftarrow$ False
                        break
                    **end**
                **end**
                **if** $new\_obj$ is True **then**
                    $dict[bb] \leftarrow next\_frames\_to\_consider$
                    $tree\_counter++$
                **end**
            **end**
        **end**
    **end**
**end**
```

Algorithm 1: Counting Algorithm. bb stands for bounding box, th for threshold and IOU for Intersection Over Union.

the counting module we obtain a tuple of the GPS coordinates of the route and the count of trees in that route. We now create the category map using the tuple and a tree count density index that we develop and describe further. We create the index empirically by analysing the number of trees in thirty routes, each of length of 1 km. We create 5 bins according to the number of trees per km. Each bin has a corresponding interpretation of the street tree coverage ranging from very low tree coverage to very good tree coverage. The index is shown in Table 2. In case, we want to analyse routes of length different than 1 km, then the index is scaled linearly.

Figure 5 shows how the category map looks like for sample routes in our training data. The advantage that the category map provides is that it helps to understand the high-level view of the number of trees in various streets of a city, town, or village. This in turn assists the policy-makers to quickly identify tree-starved roads and employ corrective measures to improve their green cover. One limitation of this map is when the tree distribution in a particular route is disparate. For instance, a route of 3 km has a total tree count of greater than 180, rendering this route a dark green. However, this route has a disparate distribution of trees such that, the first 1.5 km has 150 trees and the rest of the 1.5 km has just 30 trees. In this case, the category map would not showcase this disparity in the tree density. To solve this problem, we develop another map that we discuss in the next section, which showcases finer details about the tree densities in a route.

Table 2: Tree Count Density Index

Tree Count (per km)	Interpretation (category)	Colour on map
< 20	Very Low	Black
20 to 30	Low	Red
30 to 40	Moderate	Blue
40 to 50	Good	Green
> 50	Very Good	Dark green

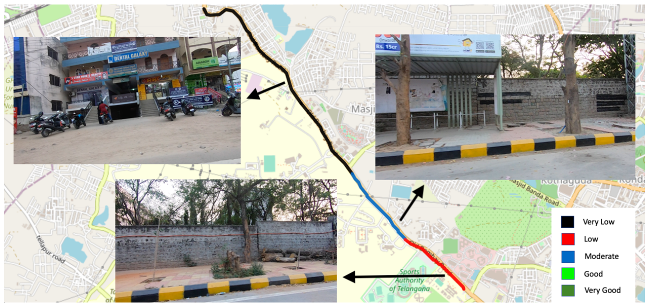

Figure 5: An example of a category map: the routes shown here are colour-coded to showcase the tree coverage in the corresponding streets. The images marked with the arrow indicate the scenes at particular instances on these routes. Note the image corresponding to the black route is tree-starved.

4.3 Kernel Density Ranking and Density Map

To interpret the results of a street tree detection and counting model, it is desirable to find out regions of high, low and moderate tree densities. Typically, Kernel Density Estimation (KDE) is used to estimate such densities. KDE allows us to estimate the probability density function from our finite dataset of detected street trees and corresponding GPS coordinates. However, KDE does not work well with GPS data [6]. Kernel Density Ranking (KDR) [6] is a better approach which is more conducive to GPS data. KDR is derived from KDE and thus we briefly explain the working of KDE first.

Let n be the total number of trees detected and $\chi = \{X_1, X_2, ..., X_n\}$ be the GPS positional data of each individual tree. The i^{th} location is $X_i = (x_i, y_i)$ where x_i and y_i denote the latitude and longitude.

Kernel Density Estimation. This approach first partitions a wider area which consists of all the GPS coordinates in χ and partitions it into a raster grid. Each cell x, in this raster grid is assigned a value based on the distances from the center of the cell to the locations in χ. The value of a grid cell x, is directly

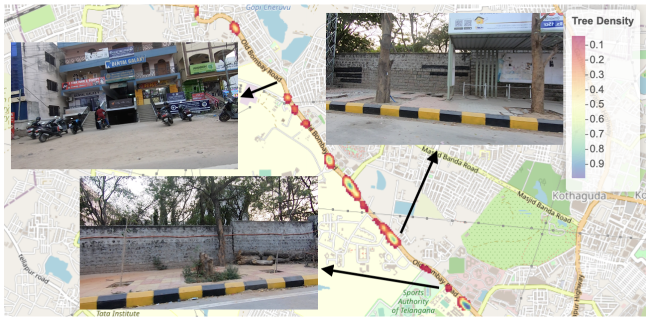

Figure 6: An example of a density map: the routes shown here are the same as in Figure 5. The density of trees correspond to the colour of the corresponding route in Figure 5. The images marked with the arrow show the real-world scene at certain instances on these routes.

proportional to the density of trees in that cell. The estimate of the tree density at a grid point x is given by

$$\hat{p}(x) = \frac{1}{nh^2}\sum_{i=0}^{n} K\left(\frac{d_i(x)}{h}\right) \tag{1}$$

Here $K()$ is a kernel function, h is the bandwidth or smoothing parameter, and $d_i(x)$ is the distance between the grid point x and the i^{th} tree GPS coordinate $X_i = (x_i, y_i) \in \chi$. The most usual choice for $K()$ is a Gaussian function. It is important to note that while calculating the tree density at a grid cell in the equation (1), we are considering the GPS points of all trees in χ. However, this is not desirable as a tree GPS point that is at a large distance from a particular grid cell x, would not affect the density of that grid cell x. Thus, we choose a kernel function from Silverman et al. [21] that caters to the aforementioned problem and the KDE becomes

$$\hat{p}(x) = \frac{3}{\pi h^2}\sum_{d_i(x)<h}\left(1 - (\frac{d_i(x)}{h})^2\right) \tag{2}$$

Thus, tree GPS locations in χ outside a circle with radius h centered at x are dropped in the evaluation of $\hat{p}(x)$. This ensures that when calculating the tree density at grid point x, the trees that are at a distance greater than h do not contribute to it.
Kernel Density Ranking. KDR is defined as:

$$\hat{\alpha}(x) = \frac{1}{n}\sum_{i}^{n} I\left(\hat{p}(X_i) \le \hat{p}(x)\right) \tag{3}$$

where $I(\omega)$ is the indicator function. The density ranking function $\hat{\alpha}(x)$ is the fraction of observations in $\chi = \{X_1, X_2, \ldots X_n\}$ whose KDE is lower than the KDE of the given point x. The density ranking function $\hat{\alpha}(x)$ is a probability-like quantity that takes values between 0 and 1. Density ranking has a straightforward interpretation. If a grid point x has a value close to 1 that means that the tree density (KDE) at that grid point is greater than the tree density (KDE) of almost all the points in χ. For instance, for a point x with $\hat{p}(x) = 0.7$, the probability density (measured by the KDE $\hat{p}$) at point x is higher than the probability density of 70% of all GPS locations in χ.

Using the kernel density ranking algorithm, we create the density map for street trees. An example density map is shown in Figure 6. The route shown in the density map in Figure 6 is the same as the route in the category map in Figure 5. We can see that the colour of the route corresponds to the tree density in that route. Further, the density map provides finer details related to the tree density. For instance, for the black route in Figure 5, the density map illustrates the exact locations of areas that are tree-starved and areas that have relatively more trees. The density map along with the category map, can help provide municipal or civic authorities with a high-level view of street tree densities of a city.

Table 3: Tree detection and counting results for various detectors. MSE is Mean Absolute Error, and TCDCA is Tree Count Density Classification Accuracy defined in Section 5.

Model	mAP	MAE	TCDCA
Faster RCNN	81.01%	6.12	74.19%
YOLOv4	82.50%	4.35	90.32%
YOLOv5s	79.29%	7.22	67.74%
YOLOv5l	**83.74%**	**3.09**	**96.77%**

5 EXPERIMENTS AND RESULTS

We discuss the results of our detection and counting experiments in this section. Finally, we also present the obtained category and density maps on some of our test videos.
Detection We train the YOLOv5l model for 100 epochs with a batch size of 4 on a GeForce GTX 1080 GPU. We use pre-trained weights on the COCO detection dataset. When training the YOLOv5l model with the default hyperparameters in the official YOLOv5 [9] codebase, we found that while the box loss on the validation set decreases, the objectness loss on the validation set reaches the lowest point and start increasing. To resolve this, we reduce the contribution of the objectness loss to the overall loss function by half. The mAP we obtain is 83.74%. We also try other models such as Faster RCNN, YOLOv4, and YOLOv5s, and none of these give a better mAP than YOLOv5l. The summary of the tree detection results for all the models is shown in Table 3. Overall, we obtain a 2.73% improvement of mAP over the baseline Faster RCNN model.
Counting. The test data for tree counting is different from that for detection as shown in Table 1. The test set for tree counting consists of 5 videos with a total of 25 km being covered. These videos are first pre-processed and converted into smaller segments as discussed in Section 4. The ground truth of the tree counts for these routes is manually calculated by humans. We measure the counting performance of our models using two metrics:

(1) Mean Absolute Error (MAE)
(2) Tree Count Density Classification Accuracy (TCDCA)

Let gt^i_{tree} and p^i_{tree} be the ground truth and predicted tree count for the i^{th} route. Let N be the total number of routes. Then,

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \parallel gt^i_{tree} - p^i_{tree} \parallel_1 \tag{4}$$

As shown in Table 3, we obtain a MAE of 3.1 with YOLOv5l and achieve an improvement of 3.0 in MAE over the baseline. This means that on average, the predicted count of trees varies from the true count of trees by an estimated value of 3.1.

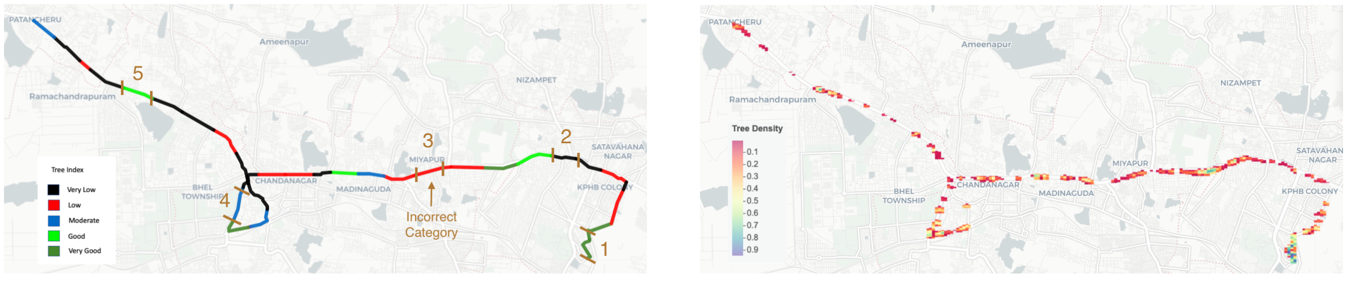

Figure 7: (a) The category map based on the routes in our test data. The numbers on this map indicate the corresponding row for this route in Table 4 and the extent of these routes are marked by the brown lines. (b) The density map based on the routes in our test data, can be compared with the category map (a) to find that the colour of the route indeed corresponds to the tree density in that route.

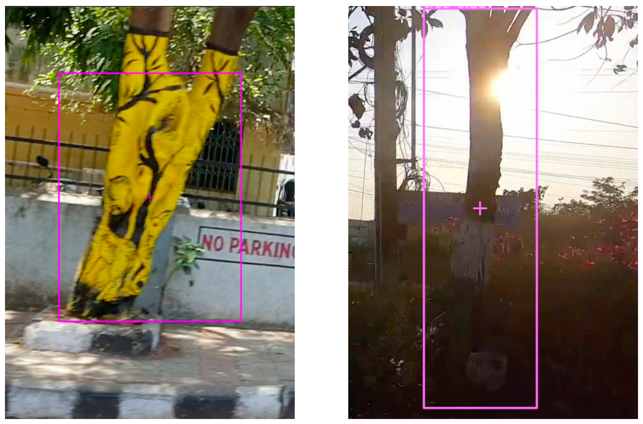

Figure 8: Interesting examples: our model detects and counts (a) a tree whose trunk has been completely painted (such an example is not present in our training data) and (b) a tree with trunk image distorted due to glare from the sun.

TCDCA measures how accurate is our system at predicting the correct category according to the tree count density index for individual streets. Let R be the number of routes with correctly predicted category and N be the total number of routes. We define the TCDCA as $\frac{R}{N}$. We obtain a TCDCA of 96.77% using our final YOLOV5l model, which is a remarkable improvement of 22.58% over the baseline Faster RCNN model. This means that our system predicts the correct category for a route with a high accuracy. The category to which a street belongs to would help municipalities and civic authorities to identify which streets are tree-starved and which streets have ample trees. This in turn will help them to quickly deploy afforestation efforts. Our system can be used to identify such streets with a high reliability and with minimal costs. The MAE and TCDCA results for other models is shown in Table 3. Some interesting results are shown in Figure 8. We strongly recommend reviewers to see the supplementary video demo showing tree detection and counting results on test samples from our dataset.

Category Map and Density Map. Figure 7 shows the resultant category and density maps for the routes that are part of our test set. We have indicated the actual count of trees for a few routes in Figure 7 (a) in Table 4. As we can see in Figure 7 (a), only one route is incorrectly classified. We can also observe that the tree density in the density map corresponds to the colour of the route in the category map. For instance, route 2 indeed has less tree density than route 1 or 5. Thus, the category map helps to quickly, affordably and reliably identify tree-starved streets, and the density map provides finer details regarding tree-starved areas in that particular street.

Table 4: Few results for the tree counts and the corresponding category according to the tree count density index. The corresponding routes on a map is shown in Figure 7 (a). All of these routes are part of our test set.

Route	Distance (km)	GT Count	Pred Count	GT Category	Pred Category
1	0.63	125	118	Very High	Very High
2	0.60	11	9	Very Low	Very Low
3	0.89	32	21	Moderate	Low
4	0.75	29	27	Moderate	Moderate
5	0.70	36	35	High	High

6 CONCLUSION

This paper presents a system for street tree detection, counting, and visualizing the tree distribution through two maps. Such a high-level visualization can help civic or municipal authorities identify tree-starved areas and employ afforestation efforts there. Specifically, we develop a unique setup and a street tree dataset with a novel annotation method, tweak and train a YOLOv5l model to detect street trees, and devise a new street tree counting algorithm. We achieve an mAP of 83.74% and a tree count density classification accuracy of 96.77%. Further, to have meaningful results, we create two maps. The first is based on a tree count density index, an index that we develop to show the tree coverage on a route using colour codes on a map according to the tree count. The second is a map to showcase street tree densities.

Future direction of this work may include 1) incorporate new species of trees into our dataset, 2) upgrade the capability of our model to detect street trees that need protection, and 3) expand the testing of our system to other cities in India as well as outside India.

Acknowledgment: We thank iHubData, the Technology Innovation Hub (TIH) at IIIT-Hyderabad for supporting this project.

REFERENCES

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-yuan Liao. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. (04 2020).

[2] Steve Branson, Jan Wegner, David Hall, Nico Lang, and Pietro Perona. 2018. From Google Maps to a fine-grained catalog of street trees. *ISPRS Journal of Photogrammetry and Remote Sensing* 135 (01 2018), 13–30. https://doi.org/10.1016/j.isprsjprs.2017.11.008

[3] Bill Cai, Xiao-jiang Li, and Carlo Ratti. 2019. Quantifying Urban Canopy Cover with Deep Convolutional Neural Networks. (12 2019).

[4] George Chen and A. Zakhor. 2009. 2D tree detection in large urban landscapes using aerial LiDAR data. *Proceedings - International Conference on Image Processing, ICIP*, 1693 – 1696. https://doi.org/10.1109/ICIP.2009.5413699

[5] Xu Chen, Qingyan Meng, Die Hu, Linlin Zhang, and Jian Yang. 2019. Evaluating Greenery around Streets Using Baidu Panoramic Street View Images and the Panoramic Green View Index. *Forests* 10, 12 (2019). https://doi.org/10.3390/f10121109

[6] Yen-Chi Chen and Adrian Dobra. 2020. Measuring human activity spaces from GPS data with density ranking and summary curves. *The Annals of Applied Statistics* 14 (03 2020), 409–432. https://doi.org/10.1214/19-AOAS1311

[7] Susan D Day, P Eric Wiseman, Sarah B Dickinson, and J Roger Harris. 2010. Tree root ecology in the urban environment and implications for a sustainable rhizosphere. *Journal of Arboriculture* 36, 5 (2010), 193.

[8] Kenta Itakura and Fumiki Hosoi. 2020. Automatic Tree Detection from Three-Dimensional Images Reconstructed from 360° Spherical Camera Using YOLO v2. *Remote Sensing* 12, 6 (2020). https://doi.org/10.3390/rs12060988

[9] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu, changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. 2020. *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*. https://doi.org/10.5281/zenodo.4154370

[10] C.M Kalansuriya, Ananda Pannila, and D.U.J. Sonnadara. 2009. Effect of roadside vegetation on the reduction of traffic noise levels. *Proceedings of the Technical Sessions* 25 (01 2009), 1–6.

[11] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. 2020. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV* (2020).

[12] Timo Lahivaara, Aku Seppanen, Jari P. Kaipio, Jari Vauhkonen, Lauri Korhonen, Timo Tokola, and Matti Maltamo. 2014. Bayesian Approach to Tree Detection Based on Airborne Laser Scanning Data. *IEEE Transactions on Geoscience and Remote Sensing* 52, 5 (2014), 2690–2699. https://doi.org/10.1109/TGRS.2013.2264548

[13] Xiaojiang Li, Chuanrong Zhang, Weidong Li, Robert Ricard, Qingyan Meng, and Weixing Zhang. 2015. Assessing street-level urban greenery using Google Street View and a modified green view index. *Urban Forestry & Urban Greening* 14, 3 (2015), 675–685. https://doi.org/10.1016/j.ufug.2015.06.006

[14] Kostas Mouratidis. 2019. The impact of urban tree cover on perceived safety. *Urban Forestry & Urban Greening* 44 (08 2019), 126434. https://doi.org/10.1016/j.ufug.2019.126434

[15] Yuchu Qin, Antonio Ferraz, Clement Mallet, and Corina Iovan. 2014. Individual tree segmentation over large areas using airborne LiDAR point cloud and very high resolution optical imagery. In *2014 IEEE Geoscience and Remote Sensing Symposium*. 800–803. https://doi.org/10.1109/IGARSS.2014.6946545

[16] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. (04 2018).

[17] Josef Reitberger, C. Schnörr, Peter Krzystek, and Uwe Stilla. 2009. 3D Segmentation of single trees exploiting full waveform LIDAR data. *Isprs Journal of Photogrammetry and Remote Sensing - ISPRS J PHOTOGRAMM* 64, 561–574. https://doi.org/10.1016/j.isprsjprs.2009.04.002

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1* (Montreal, Canada) *(NIPS'15)*. MIT Press, Cambridge, MA, USA, 91–99.

[19] Dr Roy, Jason Byrne, and Catherine Pickering. 2012. A systematic quantitative review of urban tree benefits, costs, and assessment methods across cities in different climatic zones. *Urban Forestry & Urban Greening* 11 (12 2012), 351–363. https://doi.org/10.1016/j.ufug.2012.06.006

[20] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, Ben Hoff, TOsmanov, Dmitry Kruchinin, Artyom Zankevich, DmitriySidnev, Maksim Markelov, Johannes222, Mathis Chenuet, a andre, telenachos, Aleksandr Melnikov, Jijoong Kim, Liron Ilouz, Nikita Glazov, Priya4607, Rush Tehrani, Seungwon Jeong, Vladimir Skubriev, Sebastian Yonekura, vugia truong, zliang7, lizhming, and Tritin Truong. 2020. *opencv/cvat: v1.1.0*. https://doi.org/10.5281/zenodo.4009388

[21] B. W. Silverman. 1986. Density Estimation for Statistics and Data Analysis. *Monographs on Statistics and Applied Probability* (1986).

[22] V. R. Stovin, A. Jorgensen, and A. Clayden. 2008. STREET TREES AND STORMWATER MANAGEMENT. *Arboricultural Journal* 30, 4 (2008), 297–310. https://doi.org/10.1080/03071375.2008.9747509 arXiv:https://doi.org/10.1080/03071375.2008.9747509

[23] Lionel Sujay Vailshery, Madhumitha Jaganmohan, and Harini Nagendra. 2013. Effect of street trees on microclimate and air pollution in a tropical city. *Urban Forestry & Urban Greening* 12, 3 (2013), 408–415. https://doi.org/10.1016/j.ufug.2013.03.002

[24] Porporato A Vico G, Revelli R. 2014. Ecohydrology of street trees: design and irrigation requirements for sustainable water use. *Ecohydrology* 7 (04 2014), 508 – 523. https://doi.org/10.1002/eco.1369

[25] Chien-Yao Wang, Hong-yuan Liao, Yuen-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. 2020. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. 1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203

[26] Kathleen Wolf, Sharon Lam, Jennifer McKeen, Gregory Richardson, Matilda van den Bosch, and Adrina Bardekjian. 2020. Urban Trees and Human Health: A Scoping Review. *International Journal of Environmental Research and Public Health* 17 (06 2020), 4371. https://doi.org/10.3390/ijerph17124371

[27] Qian Xie, Dawei Li, Zhenghao Yu, Jun Zhou, and Jun Wang. 2019. Detecting Trees in Street Images via Deep Learning With Attention Module. *IEEE Transactions on Instrumentation and Measurement* PP (12 2019), 1–1. https://doi.org/10.1109/TIM.2019.2958580

[28] Jun Yang, Linsen Zhao, Joe Mcbride, and Peng Gong. 2009. Can you see green? Assessing the visibility of urban forests in cities. *Landscape and Urban Planning* 91 (06 2009), 97–104. https://doi.org/10.1016/j.landurbplan.2008.12.004

[29] Lin Yang, Xiaqing Wu, Emil Praun, and Xiaoxu Ma. 2009. Tree Detection from Aerial Imagery. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Seattle, Washington) *(GIS '09)*. Association for Computing Machinery, New York, NY, USA, 131–137. https://doi.org/10.1145/1653771.1653792

[30] Ling Yao, Tang Liu, Jun Qin, Ning Lu, and Chenghu Zhou. 2021. Tree counting with high spatial-resolution satellite imagery based on deep neural networks. *Ecological Indicators* 125 (2021), 107591. https://doi.org/10.1016/j.ecolind.2021.107591

[31] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid Scene Parsing Network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6230–6239. https://doi.org/10.1109/CVPR.2017.660