Cataloging Public Objects Using Aerial and Street-Level Images – Urban Trees

Jan D. Wegner[a],[*]   Steve Branson[b],[*]   David Hall[b]   Konrad Schindler[a]   Pietro Perona[b]
ETH Zürich[a]   California Institute of Technology[b]

Abstract

Each corner of the inhabited world is imaged from multiple viewpoints with increasing frequency. Online map services like Google Maps or Here Maps provide direct access to huge amounts of densely sampled, georeferenced images from street view and aerial perspective. There is an opportunity to design computer vision systems that will help us search, catalog and monitor public infrastructure, buildings and artifacts. We explore the architecture and feasibility of such a system. The main technical challenge is combining test time information from multiple views of each geographic location (e.g., aerial and street views). We implement two modules: det2geo, which detects the set of locations of objects belonging to a given category, and geo2cat, which computes the fine-grained category of the object at a given location. We introduce a solution that adapts state-of-the-art CNN-based object detectors and classifiers. We test our method on "Pasadena Urban Trees", a new dataset of 80,000 trees with geographic and species annotations, and show that combining multiple views significantly improves both tree detection and tree species classification, rivaling human performance.

1. Introduction

In this very moment thousands of geo-tagged images of almost any location of the populated world are being captured and shared on the web. There are two main sources of publicly available images, user-contributed photographs and imagery from online mapping services. While user-provided photographs cover mostly popular sites, systematic commercial efforts provide a homogeneous and dense coverage of the populated parts of the world, especially urban areas. This includes overhead imagery captured by satellite and aircraft, and high-resolution ground panoramas that are regularly distributed along the road network [3]. Browser-based interfaces such as Google Maps provide free and well-structured access to this rich, up-to-date and geocoded treasure trove.

Publicly available imagery has already found its use in

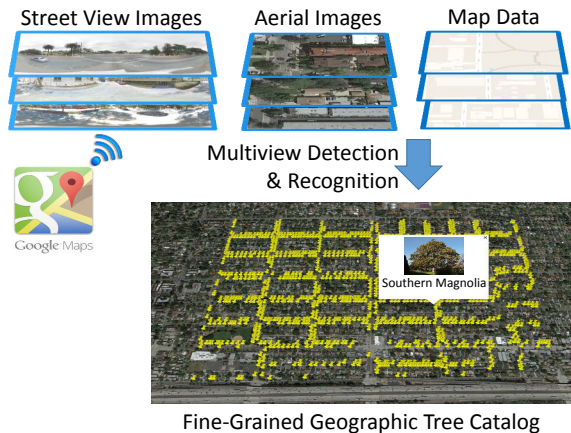[*]joint first authorship

Figure 1. Overview of proposed automated public tree cataloguing system from online maps. Aerial images and street view panoramas along with semantic map data are downloaded for some geographical region. Category detection and fine-grained classification algorithms are trained from human-annotated exemplars. Detection, classification and geolocation information is computed automatically from multiple street view images and aerial images and combined with map data to achieve a geolocated fine-grained catalog. The image shows a catalog of location and species of trees in a medium-sized city.

a great number of applications and circumstances. To cite a few: navigation and geo-localization [19, 3, 34], virtual tourism [2], urban planning and evaluation of the quality of public spaces [23, 18, 17]. However, the process of cataloguing and classifying visible objects in the public space (e.g. street signs, building facades, fire hydrants, solar panels and mail boxes) is still carried out 'by hand', often by in-person inspection or from expensive ad-hoc imagery such as LiDAR. Due to the cost, time, and organizational headache it involves, such information is rarely collected and analyzed. Harvesting such information automatically from online maps will provide inexpensive ready-to-use and reliable information to the public, to administrators, and to scientists which would greatly improve the quality and timeliness of public resource management.

We present a vision-based system that systematically detects and classifies publicly visible objects. Overhead and street-view imagery are combined to populate and update a

public inventory of trees with GPS position and fine-grained species at virtually no cost. Our methods were motivated by a large-scale tree mapping project called *Opentreemap*[1] aiming to build a centralized, publicly available, and frequently updated tree inventory for each city in the world. The project is stifled by the significant amount of human labor required to catalogue trees. We speculated that Computer Vision may make it viable and explored the question of which combination of geometry and recognition would be most appropriate. Our main contributions are:

1. det2geo: a method to generate a geographic catalog of objects belonging to a given category using multiple aerial and street-level views of each location.

2. geo2cat: a method to compute the fine-grained class label of the 3D object at a given geographical coordinate using multiple aerial and street-level views.

3. Pasadena Urban Trees: A dataset of about 80,000 trees tagged with species labels and geographic locations, along with a comprehensive set of aerial, street view, and map images downloaded from Google Maps (>100,000 images).

To build *geo2cat* and *det2geo*, we created methods to automatically download and mutually register aerial and street view images from Google maps. We document the appropriate geometric routines needed to register each type of Google maps image, such that they can easily be integrated with computer vision algorithms (Section 3). We believe that, compared to most prior work, we have gone more in depth to integrate modern, learning-based methods for detection (Section 4) and recognition (Section 5) with multi-view geometry and maps data to obtain *multi-view visual detection and recognition*. We find that multi-view recognition of 3D objects provides significant empirical gains over the customary single view approach: mean average precision increases from 42% to 71% for tree detection, and tree species recognition accuracy is improved from 70% to 80% (Section 7). We motivate and test our algorithms with an important real life application and a new dataset (Section 6). Our methods are already working well enough to have practical impact.

2. Related work

During the past ten years a number of creative and potentially useful ideas have emerged, how to make use of publicly available geo-referenced imagery. Amongst these are the analysis of social networks [7], the identification of popular landmarks [29, 8], scene reconstruction, 3D models and visualizations [20, 2, 1, 11] and 4D models that capture changes over time [35]. Many of these studies are based on images shared by individual users, whose uneven spatial distribution has been recognized as a fundamental limitation [20, 2]. Regularly sampled street-level and satellite pictures have been used to obtain more complete coverage

and reconstructions [45]. Researchers have proposed visual recognition and classification methods for inferring Geo-localization from single images [19, 30, 31] for applications like land cover classification [28] or to build large-scale maps of snow coverage or bird species distribution [46].

A number of studies have proposed methods for automating the detection of publicly visible objects. These methods make use of ad-hoc special-purpose imagery [42] or laser scans [16, 26]. One recent approach to tree detection in cities with aerial RGB images is [48]. They first classify aerial images into tree and background pixels with a CRF under the standard Potts prior. Single trees are extracted by matching a template to candidate tree regions, followed by a set of rules that greedily selects best matches while minimizing overlap of adjacent templates. It is not yet clear whether that method will scale up to entire cities with many different tree shapes since the experiments are carried out on limited datasets. The study focusses on detection and does not address species classification.

Tree species classification from remote sensing data usually relies either on species-specific spectral signatures in hyperspectral data [6, 39] or on dense full-waveform LiDAR returns that capture the distinctive reflectance patterns of the laser beam penetrating the canopy [5, 49]; or on a combination of LiDAR data and aerial imagery, to exploit both the height distribution of the LiDAR returns and the image radiometry and texture [21, 24, 22]. Classifiers are mostly trained for a relatively small number of species (3 in [27, 21, 24], 4 in [22], 7 in [47, 37]).

An alternative to remote sensing is to acquire images of tree details (e.g., of leafs, bark) in situ, and match them to a reference database [9, 25, 36, 15, 14]. If turned into a smart-phone app like *Pl@ntNet* [15, 14] or *Leafsnap* [25] they enable anyone to recognize the species of a particular plant. The main goal of such apps has been to educate users about plants. It seems difficult to collect a complete and homogeneous tree inventory with them due to the fact that each tree must be visited by at least one person.

Recent work tries to establish correspondence between street-view data and oblique aerial imagery with a learned matching function [32, 33]. We are not aware of any prior work that combines aerial and street view images as different cues that can be used with modern learning-based detection and fine-grained recognition algorithms. We also do not know of any work that recognizes more than a handful of species without dedicated sensor data like hyper-spectral images or high-density LiDAR.

Unlike previous studies we approach the detection and classification of urban objects, trees in this paper, by using exclusively images that are publicly available. We find that the two points of view, aerial and street-view, complement each other well. The trick is to do late fusion of category labels: the outputs of state-of-the-art CNN detectors and clas-
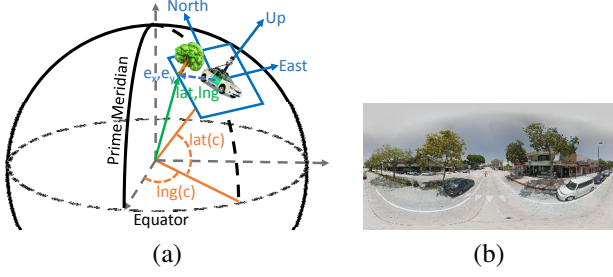
Figure 2. **(a)** Geometry of the street view acquisition system. The Google car sits on the surface of the earth at coordinate (lat,lng). A tree is represented in ENU coordinates formed by a plane tangent to the earth at the location of the camera. The heading of the car rotates this point to determine the tree's location in a panorama image. **(b)** An example of a $360°$ street view panorama image.

sifiers are combined in a probabilistic framework. In this way, one circumvents the difficult problem of establishing sparse (let alone dense) correspondence across very wide ($\approx 90°$) baselines and scale differences. Note also that, unlike most other methods, our formulation does not require any prior segmentation into superpixels, hierarchies of ad-hoc rules, or pre-designed top-down tree models. In many cases it is not even necessary to annotate training data, because geo-referenced tree inventories already exist in many regions of the world – *i.e.*, our training data was generated by downloading publicly available resources from the web.

3. Online map data

As a data source we use publicly available images of Google maps, including aerial imagery, street view imagery, and map data (see Figure 1). Given a geographic region of interest (*e.g.*, the city of New York), we first densely download all relevant images from static URLs. For each type of image modality v (*e.g.*, street view or aerial view), we computed the function $\ell' = \mathcal{P}_v(\ell, c)$ that projects a geographic latitude/longitude location $\ell = (\text{lat}, \text{lng})$ to its corresponding image location $\ell' = (x, y)$ given camera parameters c. These projection functions will provide a building block for using Google maps data with different types of computer vision algorithms in subsequent sections of the paper.

Street view images: We can estimate geographic coordinates of an object from a single street view panorama under the assumption of known camera height and locally flat terrain. We first represent the object in Local east, north, up (ENU) coordinates with respect to the position of the camera. This means that if we position a plane tangent to the surface of the earth at $\text{lat}(c)$, $\text{lng}(c)$ and define a coordinate system where the x-axis points east, the y-axis points north, and the z-axis points up (Fig. 2), then the ENU position of

an object sitting on the ground at (lat, lng) is

$$
\begin{aligned}
(e_x, e_y, e_z) = \big(&R\cos[\text{lat}(c)]\sin[\text{lng} - \text{lng}(c)], \\
&R\sin[\text{lat} - \text{lat}(c)], -h\big)
\end{aligned}
\tag{1}
$$

where h is the height that the Google street view camera is mounted above the ground and R is the radius of the earth. The object is then at a distance $z = \sqrt{e_x^2 + e_y^2}$ from the camera (measured on the ground plane). It sits at a clockwise angle of $\arctan(e_x, e_y)$ from north, and a tilt of $\arctan(-h, z)$ (Fig. 2). The ENU coordinate can be converted into cylindrical coordinates using the camera's heading to obtain image coordinates $\ell' = (x, y)$. The resulting image projection $(x, y) = \mathcal{P}_{sv}(\text{lat}, \text{lng}, c)$ is computed as

$$
\begin{aligned}
x &= (\pi + \arctan(e_x, e_y) - \text{yaw}(c))\, W/2\pi \\
y &= (\pi/2 - \arctan(-h, z))\, H/\pi
\end{aligned}
\tag{2}
$$

where the panorama image is $W \times H$ pixels.

Aerial images: Due to space limitations, we include the form of $\ell = \mathcal{P}_v^{-1}(\ell', c)$ and further information about geometric transformations in the supplementary results.

4. det2geo: Multi-view detection

The goal of det2geo is to process image sets and map layers downloaded from Google maps and automatically generate a catalog of all geographic locations of an object of interest. We introduce methods to augment state-of-the-art learning based object detection systems with multi view geometry and maps such as the location of roads.

A minor complication to using conventional object detection methods is that our target outputs and training annotations are geographic coordinates (latitude/longitude)–they are points rather than bounding boxes. A simple solution is to interpret boxes as regions of interest for feature extraction rather than as physical bounding boxes around an object. At train time we can convert geographic coordinates to pixel coordinates using the appropriate projection function $\mathcal{P}_v(\ell, c)$ and create boxes with size inversely proportional to the distance of the object to the camera. At test time, we can convert the pixel location of the center of a bounding box back to geographic coordinates using $\mathcal{P}_v^{-1}(\ell', c)$. Doing so makes it possible to train single-image detectors. In the next section, we show how to build a multi-view detector that combines multiple images and other sources of information probabilistically.

4.1. Multi-view detection

As a base detection system, we use the publicly available implementation of Faster R-CNN [38]. Faster R-CNN is a recent state-of-the-art method that significantly improves the speed of R-CNN [13] and Fast R-CNN [12], all of which are based on convolutional neural networks (CNNs) and region proposals.
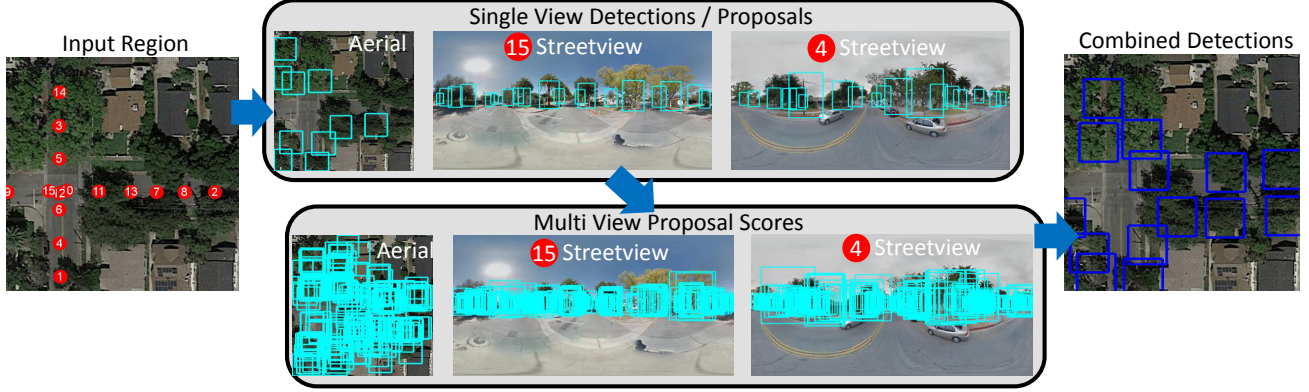
Figure 3. **Multi View Detection:** We begin with an input region (left image), where red dots show available street view locations. Per view detectors are run in each image (top middle), and converted to a common geographic coordinate system. The combined proposals are converted back into each view (bottom middle), such that we can compute detection scores with known alignment between each view. Multi-view scores are combined with semantic map data and spatial reasoning to generate combined detctions (right).

In our approach, we allow promising detection regions in one view to augment the region proposal set of the other views. The multiview detection score of a geographic coordinate is obtained by combining the corresponding detection scores in each view, and thresholding and non-maximal suppression occurs over regions represented in geographic coordinates rather than in pixel coordinates of any one view. We use the following procedure:

1. For each view v, generate region proposals R_v by running a detector with a liberal detection threshold

2. Compute a combined multi view region proposal set R by taking the union of all view proposals R_v after warping them into geographic coordinates $R = \{\mathcal{P}_v^{-1}(\ell_{vj}, c_v)\}_{j=1}^{|R_v|}$, where ℓ_{vj} is the pixel location of the jth region center.

3. For each view v, evaluate detection scores on the combined multi view proposal set R after converting each region ℓ_k into image coordinates $\mathcal{P}_v(\ell_k, c)$.

4. Compute a combined detection score by adding together the detection scores of each view. Apply a detection threshold τ_2 and suppress overlapping regions to obtain geographic detections.

Figure 3 shows a visualization of the approach. It is designed to be able to always combine information from each view, even when the region proposal or detection system fails in a subset of the views. Additionally, we attempt to minimize computation time by keeping the combined proposal set R as small as possible. Note that although we use Faster R-CNN, our method can work with any major object detection algorithm, including methods that use region proposals or methods that compute detection scores in sliding window fashion. A limitation though is that simply adding the detection scores together is suboptimal when

some views are more reliable sources of information than others. In the next section, we describe a procedure to learn how to combine them probabilistically and also include other sources of information.

4.2. Probabilistic model

Let T be a candidate set of object detections, where each $t_i \in T$ represents an object location in geographic coordinates. Let $\mathrm{lat}(t)$ and $\mathrm{lng}(t)$ be shorthand for the latitude and longitude of t. Our goal is to choose the best set of objects T that factors in different sources of information, including aerial view imagery, street view imagery, semantic map data (*e.g.*, the location of roads), and spatial context of neighboring objects. We combine these sources of information using a conditional random field:

$$\log p(T) = \sum_{t \in T} \bigg(\underbrace{\Lambda(t, T; \alpha)}_{\text{spatial context}} + \underbrace{\Omega(t, \mathrm{mv}(t); \beta)}_{\text{map image}}$$
$$+ \underbrace{\Psi(t, \mathrm{av}(t); \gamma)}_{\text{aerial view image}} + \sum_{s \in \mathrm{sv}(t)} \underbrace{\Phi(t, s; \delta)}_{\text{street view images}} \bigg) - Z \quad (3)$$

where $\Lambda()$, $\Omega()$, $\Psi()$, and $\Phi()$ are potential functions with learned parameters α, β, δ, γ, $\mathrm{av}(t)$ and $\mathrm{mv}(t)$ are the IDs of aerial and map view images that contain object t, $\mathrm{sv}(t)$ is the ID of the set of street view images where t is visible (with associated meta data defining the camera position), and Z is a normalization constant. We define these terms below:

Aerial View Potential: We define the aerial view potential to be the detection score evaluated at the appropriate region:

$$\Psi(t, \mathrm{av}(t); \gamma) = \mathrm{CNN}\left(X(\mathrm{av}(t)), \mathcal{P}_{av}(t); \gamma\right) \quad (4)$$

where $X(\mathrm{av}(t))$ is the aerial image, γ encodes the weights of the aerial view detection CNN, and $\mathcal{P}_{av}(t)$ transforms

between pixel location and geographic coordinates. See supplementary material for details.

Street View Potential: Similarly, we define the potential function for a street view image $s \in \text{sv}(t)$ as

$$\Phi(t, s; \delta) = \text{CNN}\left(X(s), \mathcal{P}_{sv}(t, \text{c}(s)); \delta\right) \qquad (5)$$

where $X(s)$ is a street view image, δ encodes the weights of the street view detection CNN, and $\mathcal{P}_{sv}(t, c)$ is defined in Equation 2. Note that each object t might be visible in multiple street view images. We tried two approaches for defining the set $\text{sv}(t)$ of relevant images: 1) We select a single street view image that is closest to the proposed object location t, or 2) We select all images that were taken within a prespecified distance threshold τ_{sv} between t and the camera location $\text{c}(s)$. We empirically found the first approach to give better results, probably due to lower likelihood of occlusion and effect of camera heading error[2].

Spatial Context Potential: The purpose of the spatial context potential is to impose a prior on the distance between neighboring objects. For example, two trees cannot physically grow in the same location and are unlikely to be planted in very close proximity. At the same time, neighboring trees are often planted in regularly spaced intervals parallel to the road. Let $d_s(t, T) = \min_{t' \in T} \|t - t'\|^2$ be the distance to the closest neighboring object, and $Q_s(d_s(t, T))$ be a quantized version of d_s. That is, $Q_s()$ is a vector in which each element is 1 if d_s lies within a given distance range and 0 otherwise. We then learn a vector of weights α (see Fig.8 in supplementary material), where each element α_i can be interpreted as the likelihood that the closest object is within the appropriate distance range. Thus

$$\Lambda(t, T; \alpha) = \alpha \cdot Q_s(d_s(t, T)) \qquad (6)$$

In our experiments, we compare this to a term that forbids neighboring objects to be closer than τ_{nms}

$$\Lambda_{nms}(t, T; \alpha) = \begin{cases} -\infty & \text{if } d_s(t, T) < \tau_{nms} \\ 0 & \text{otherwise} \end{cases} \qquad (7)$$

This is analogous to a traditional non-maximal suppression term that suppresses overlapping bounding boxes. The learned approach has the advantage that it can learn to softly penalize objects from being too close. It can also learn that it is unlikely for an object such as a tree to be completely isolated from other trees.

Map Potential: Google maps offer additional semantic information that may provide useful priors for detection. Intuitively, an object such as a tree cannot lie in the middle of the road. Moreover, trees are often planted alongside roads at a fixed distance. We download Google maps images and

use simple image processing techniqes to compute the distance from each pixel to the nearest road[3]. Let $d_m(t)$ be the distance in meters between an object t and the closest road. Similar to the spatial context term, we quantize this distance into geometrically increasing intervals and learn a prior β_i (see Fig.8 in supplementary material) on each interval:

$$\Omega(t, \text{mv}(t); \beta) = \beta \cdot Q_m(d_m(t)) \qquad (8)$$

Inference: At test time, our goal is to choose a catalog of object detections $T^* = \arg\max_T \log(p(T))$ that maximizes Equation 3. This is in general a challenging problem; however, a widely used procedure is to iteratively add new detections using a greedy algorithm. That is, we begin with $T = \emptyset$, and iteratively append a new detection

$$t' = \arg\max_t \log(p(T \cup t)) \qquad (9)$$

stopping when no new object can be found that increases $\log p(T)$. This is efficient to compute because we can pre-compute the combined detection score $\Omega(t, \text{mv}(t); \beta) + \Psi(t, \text{av}(t); \gamma) + \sum_{s \in \text{sv}(t)} \Phi(t, s; \delta)$ for each location t in our combined multi view region proposal set R, then update our computation of the spatial term $\Lambda(t, T; \alpha)$ every time we add a new detection t'. This greedy procedure is a very commonly used procedure in object detection and is a well known probabilistic interpretation of non-maximal suppression that has known approximation guarantees for some choices of $\Lambda()$ [4, 43].

Learning: At training time, our goal is to learn parameters $\alpha^*, \beta^*, \delta^*, \gamma^* = \arg\max_{\alpha, \beta, \delta, \gamma} \log(p(T))$ that maximizes Equation 3, where T is the set of objects in our training set. For practical reasons, we use piecewise training, which is known to work well for these types of CRFs [41] and offers convenience in terms of optimization and modularity. Here, we subdivide the training set into a validation set $\mathcal{D}_v$ and training set $\mathcal{D}_t$, then learn each parameter vector $\alpha, \beta, \delta, \gamma$ separately over their respective potential terms. Next, we learn a weighted combination of each potential term on the validation set. For details see the supplementary material.

5. geo2cat: Fine-grained classification

geo2cat aims to predict the fine-grained category of an object that has already been geolocated (*e.g.*, using det2geo). We propose to apply state-of-the-art CNNs, and combine their outputs on aerial and street view imagery.

The method first obtains cropped versions of each object at different zoom levels using the appropriate projection function $\mathcal{P}_v(\ell, c)$ defined in Section 3. Each cropped

---

[2]Future extensions could better exploit multiple views by inversely weighting their influence with distance from the object, for example.

[3]Roads are distinguishable as pixels with value 255. Morphological opening removes other small symbols that also have value 255. Morphological closing removes text written on roads. A distance transform computes per pixel distances to road.

region is then fed through a CNN feature extractor. After testing several models we found that the GoogLeNet CNN model [44] offered the best compromise in terms of recognition performance, run-time, and memory consumption. We train one CNN model per viewpoint and zoom level using a log-logistic loss via stochastic gradient descent. We fine-tune the weights of each model after initializing them to weights pre-trained on ImageNet [40]. The learning rate is initially set to 0.001. After every ten epochs, it is decreased by a factor ten for 30 epochs in total. We then discard the top, fully-connected layer per model and extract features from the $pool/7 \times 7$ layer of the GoogLeNet model. The resulting feature vector per model has 1024 dimensions. We concatenate all feature vectors of all models (views and zooms) per tree to a single feature vector[4] which we then use to train a standard linear SVM[5].

6. The Pasadena Urban Trees Dataset

We apply det2geo and geo2cat to a new dataset, motivated by a collaboration with Opentreemap–a large-scale project to build and maintain a geographic catalog of tree species. We collected the dataset by downloading publicly available aerial and street view images from Google Maps at city-scale. As test area, we chose Pasadena because 1) an up-to-date tree inventory (as of 2013) with annotated species is publicly available and 2) image data are as of October 2014 (street view) and March 2015 (aerial images). The Pasadena tree inventory is publicly available as a kml-file that contains rich information for $\approx 80,000$ trees on public ground. We estimate that these constitute $\approx 20\%$ of all trees in Pasadena. Figure 4 (top) shows an overview of all trees in the Pasadena city center that were used for experiments. Each tree is mapped with its geo-location, street address, species, and trunk diameter.

Detection data set: We densely downloaded all street view, aerial, and map images for Pasadena. This included 1) 46,321 street view panorama images of size $1664 \times 832\ px$ and their associated camera locations and meta data, 2) 28,678 aerial view images of size $256 \times 256\ px$ (at $\approx 0.15\ m$ resolution), and 3) 28,678 map view images of size $256 \times 256\ px$. We converted the geographic locations of the $80,000$ Pasadena trees to the appropriate pixel locations in each image. Since the inventory does not include trees on private land, we densely labeled all tree locations in a subset of 1,000 aerial view images and 1,000 street view images using Mechanical Turk, which we used to train object detectors.

Species recognition data set: We downloaded four dif-

---

[4]to form a vector of 4096 dimensions in our case with one aerial image and street views at three different zoom levels per tree. A (probably more elegant) alternative to simple feature concatenation would be to explicitly encode the three panorama zoom levels in a single CNN architecture.

[5]experiments with Neural Nets decrease performance by 2 percent

Figure 4. Top: Overview of the Pasadena 2013 public tree inventory data set. Bottom: Aerial image and street view panorama examples from Google maps at zoom levels 40, 80, and 110.

ferent images per tree from Google Maps around the appropriate geographic position for 18 different species (see Fig. 5, 5205 trees in total) that have between 100 and 600 instances: one aerial image and street view images at three different zoom levels 40, 80, and 110 (Fig. 4 (bottom)). While automated downloads facilitated data collection for thousands of trees within a few hours, the images are subject to some noise (*e.g.*, a tree may be occluded by a truck, or it has been removed after the inventory date). Manual evaluation of a dataset subset showed that $< 5\%$ of images were affected. We did not manually filter data, so as to keep the processing pipeline fully automatic. Rather, we rely on the learning algorithm to cope with label noise.

7. Experiments

We evaluate the proposed approach in terms of detection accuracy and species classification accuracy separately on the dataset described in Section 6. We split the dataset into 16 geographically separated rectangular regions (9 for training, 6 for testing, and 1 for validation).

7.1. det2geo: Tree detection

Evaluation: We evaluated detection performance in terms of average precision (precision averaged over all levels of recall), which is the standard metric used in the VOC Pascal Detection Challenge [10]. Here, candidate trees were ranked by their score combining aerial, streetview, and map imagery and spatial context (the 4 terms in Eq. 3 for a given tree t) and enumerated in order[6]. Since our goal is to pre-

---

[6]A current limitation of the system is that it does not detect objects at the wrap-around of street view panoramas, which could be fixed by padding images from the other side of the panorama.

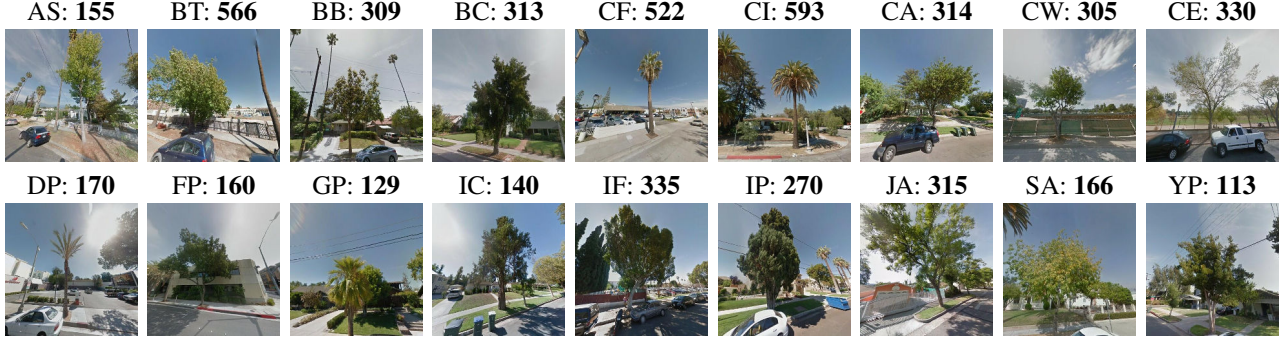| AS: **155** | BT: **566** | BB: **309** | BC: **313** | CF: **522** | CI: **593** | CA: **314** | CW: **305** | CE: **330** |
| DP: **170** | FP: **160** | GP: **129** | IC: **140** | IF: **335** | IP: **270** | JA: **315** | SA: **166** | YP: **113** |

Figure 5. Examples of the 18 species for GPS2Cat and their number of occurrences (5205 trees in total, see abbreviations in Fig. 6).

dict the latitude and longitude of trees rather than bounding boxes, predictions within a threshold of 4 meters from ground truth were considered to be valid matches. Note that the typical difference of our system from ground truth is $1-2\ m$, equal to ground truth accuracy[7]. We plot our results in Figure 6 and summarize our results below:

Significant improvement combining multiple views: Our full model obtained .706 mAP, a significant gain over the .42 mAP achieved from a pure aerial view detector using the same Faster R-CNN detection system [38]. This baseline is perhaps the most straightforward way to apply a current state-of-the-art detector to Google maps imagery without developing a way of combining multiple views. A pure street view detector achieves better performance (.581 mAP). This is a stronger baseline because it requires using geometry to combine multiple street view images–we implemented it by omitting non-streetview terms from Eq. 3 and applying non-maxima suppression (Eq. 7). We found that many penalized detections were in fact trees located on private land, which weren't included in our inventory[8]. Thus performance in practice was better than what .706 mAP would indicate.

Each component of the model is useful: To validate our model, we performed additional lesion studies. In Figure 6 top, "No Aerial", "No Streetview", and "No Map" remove the applicable potential term from the full model in Eq. 3. "No Spatial" replaces the learned spatial context term (Eq. 8) with a more conventional non-maximal suppression term (Eq. 7). We see the biggest loss in performance if we drop street view images (.706 → .462 mAP) or aerial images (.706 → .619 mAP). Dropping the map term results in a smaller drop in performance (.706 → .667 mAP). Replacing the learned spatial context potential with non-maximal suppression results in only a small drop (.706 → .69 mAP). For each lesioned version of the system we re-learn an ap-

propriate weight for each potential function on the validation set. The method "No CRF Learning" shows results if we use the full model but omitted learning these scaling factors and set them all to 1 (results in a .706 → .66 mAP drop). Additional analysis, visualizations, and qualitative examples are included in the supplementary material.

7.2. geo2cat: Tree species classification

First, we compare single view recognition (aerial or street view) to the combination of all four images per tree. If classifying tree species based on only one image per tree (instead of three zoom levels and one aerial view), we achieve ≈ .66 average precision for aerial images, and ≈ .70 per zoom level 40, 80, and 110. Combining features of all four models per tree, we see a significantly higher performance of .80 average precision and .79 average recall over all species. According to collaborators at TreePeople[9], our recognition performance is comparable to that achievable using citizen scientists, due to the significant amount of expertise required.

Close inspection of per species results (Fig. 6) reveals that not all tree species can be recognized equally well. Shamel Ashes (SA) are the most error prone (.46 precision, .35 recall) whereas American Sweetgums are recognized almost perfectly (1.0 precision, .93 recall). Note that the number of occurrences per tree is quite unevenly distributed (only 113 Yew Pines vs. 593 Canary Island Date Palms). Generally, strongly varying lighting conditions (cf. Fig. 5), partial occlusions, and differing size, shape, and general appearance per species make fine-grained classification challenging. We also observe that some species tend to be located in a few larger clusters in very different contexts (e.g., Shamel Ashes), while others are evenly distributed across the city. This makes generalization challenging.

We visualize the confusion matrix of tree species recognition in Fig. 7. We see that most tree species are recognized well (dominant, most orange values on main diagonal). We can also observe that there is hardly any dominant confusion

---

[7]We are currently initiating a field campaign with high-accuracy dGPS to quantify these errors.

[8]This is a limitation of the current ground truth and obtaining public/private land boundaries is an important next step.

[9]www.treepeople.org

Tree Detection

Full Model (0.706AP)
No Spatial (0.690AP)
No Map (0.667AP)
No Aerial (0.619AP)
No Streetview (0.462AP)
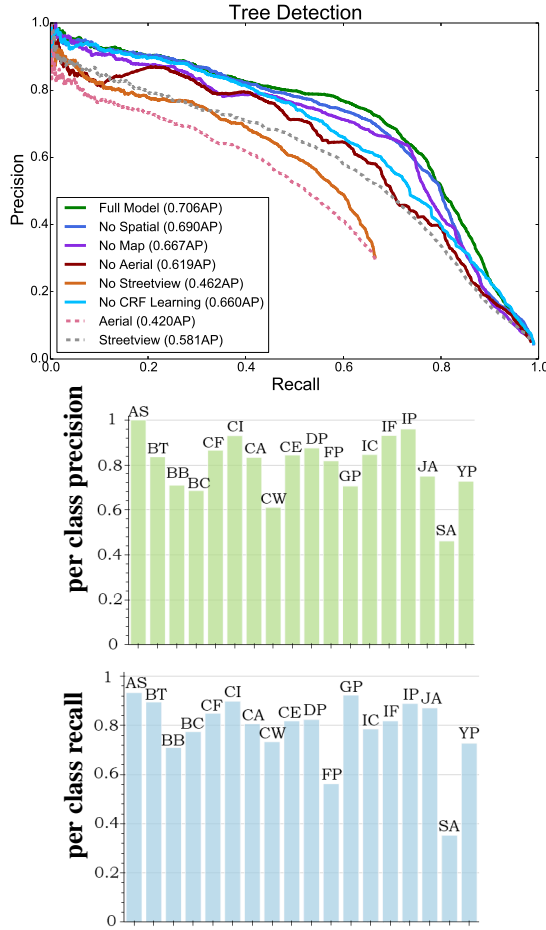No CRF Learning (0.660AP)
Aerial (0.420AP)
Streetview (0.581AP)

Figure 6. Top: Comparison of the full tree detection model to single view aerial and street view detectors and lesioned models. center, bottom: Tree species recognition results left to right: American Sweetgum (AS), Bottle Tree (BT), Brisbane Box (BB), Brush Cherry (BC), California Fan Palm (CF), Canary Island Date Palm (CI), Carob (CA), Carrotwood (CW), Chinese Elm (CE), Date Palm (DP), Fern Pine (FP), Guadalupe Palm (GP), Incense Cedar (IC), Indian Laurel Fig (IF), Italian Cypress (IP), Jacaranda (JA), Shamel Ash (SA), Yew Pine (YP).

between two particular species for any possible combination. For example, out of all tree species Shamel Ash (SA) has highest confusion with other species, but confusion is more or less evenly distributed across alternate species.

8. Conclusion

The picture of all that may be visible outdoors in the populated world is sampled with increasing temporal and spatial resolution. This 'World-Wide Light Field (WWLF)' allows machines to discover, catalogue and monitor public objects in the real 3D world. We built and tested two primitives that help automate the exploration of what is visible in the WWLF: **geo2cat** given GPS coordinates it computes
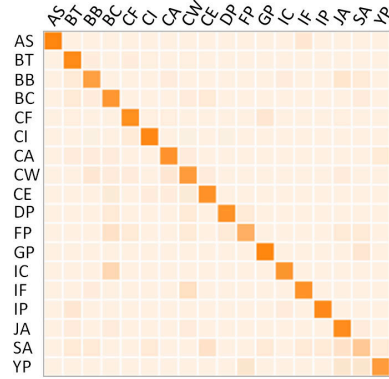

Figure 7. Confusion matrix of tree species recognition results. More orange indicates higher values (see Fig. 6 for abbreviations)

the fine-grained class of the object at that location, while **det2geo** produces the list of GPS coordinates of objects that belong to a chosen category.

We have tested our algorithms on a specific benchmark: detecting trees of the urban forest and classifying their species. geo2cat distinguishes 18 different species using state-of-the-art CNNs on RGB aerial and street view images at multiple zooms. det2geo finds the locations of urban trees (on public land), with the help of probabilistic CRF-based fusion on CNN detector scores across views.

Our experiments suggest that publicly available imagery supports both accurate detection, and accurate fine-grained classification of publicly visible objects. This is good news because cataloguing of publicly visible objects is currently carried out with specialized imagery (LiDAR, hyperspectral) that is collected ad-hoc, and/or with in-person visits.

Our next goal is to explore how well our algorithms scale to planet-wide exploration. We are planning to engage in a US-wide tree catalog of the urban forest, which is highly valuable for city planners. We will also attempt to estimate further parameters like the trunk diameter of trees. Another interesting challenge will be to combine automated methods, such as geo2cat and det2geo with crowdsourcing to fill in missing objects. To this end we will explore how to engage citizen scientists to carry out image-based and in-person verification of the data.

Our method is not limited to trees, and we expect it to generalize to other types of urban objects, for example, lamp posts, mailboxes, traffic lights. And it should become even more relevant as more city-scale imagery becomes available (e.g., videos from driver assistance systems).

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105–112, 2011.

[2] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a Day. In *International Conference on Computer Vision*, 2009.

[3] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: Capturing the world at street level. *Computer*, (6):32–38, 2010.

[4] M. Blaschko. Branch and bound strategies for non-maximal suppression in object detection. In *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2011.

[5] T. Brandtberg. Classifying individual tree species under leaf-off and leaf-on conditions using airborne lidar. *ISPRS Journal of Photogrammetry and Remote Sensing*, 61:325–340, 2007.

[6] M. L. Clark, D. A. Roberts, and D. B. Clark. Hyperspectral discrimination of tropical rain forest tree species at leaf to crown scales. *Remote Sensing of Environment*, 96:375–398, 2005.

[7] D. J. Crandall, L. Backstrom, D. Cosley, S. Suri, D. Huttenlocher, and J. Kleinberg. Inferring social ties from geographic coincidences. *Proceedings of the National Academy of Sciences*, 107(52):22436–22441, 2010.

[8] D. J. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg. Mapping the world's photos. In *Proceedings of the 18th international conference on World wide web*, pages 761–770. ACM, 2009.

[9] J.-X. Du, X.-F. Wang, and G.-J. Zhang. Leaf shape based plant species recognition. *Applied Mathematics and Computation*, 185:883–893, 2007.

[10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[11] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a Cloudless Day. In *European Conference on Computer Vision*, pages 368–381, 2010.

[12] R. Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[14] H. Goëau, P. Bonnet, A. Joly, A. Affouard, V. Bakić, J. Barbe, S. Dufour, S. Selmi, I. Yahiaoui, C. Vignau, D. Barthélémy, and N. Boujemaa. Pl@ntNet mobile 2014: Android port and new features. In *ACM International Conference on Multimedia Retrieval*, 2014.

[15] H. Goëau, P. Bonnet, A. Joly, V. Bakić, J. B. I. Yahiaoui, S. Selmi, J. Carré, D. Barthélémy, N. Boujemaa, J.-F. Molino, G. Duché, and A. Péronnet. Pl@ntNet mobile app. In *ACM International Conference on Multimedia*, 2013.

[16] A. Golovinskiy, V. Kim, and T. Funkhouser. Shape-based Recognition of 3D Point Clouds in Urban Environments. In *International Conference on Computer Vision*, 2009.

[17] K. Hara, S. Azenkot, M. Campbell, C. L. Bennett, V. Le, S. Pannella, R. Moore, K. Minckler, R. H. Ng, and J. E. Froehlich. Improving public transit accessibility for blind riders by crowdsourcing bus stop landmark locations with google street view: An extended analysis. *ACM Transactions on Accessible Computing (TACCESS)*, 6(2):5, 2015.

[18] K. Hara, V. Le, and J. Froehlich. Combining crowdsourcing and google street view to identify street-level accessibility problems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 631–640. ACM, 2013.

[19] J. Hays and A. Efros. im2GPS: estimating geographic information from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[20] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26(3):4, 2007.

[21] V. Heikkinen, I. Korpela, T. Tokola, E. Honkavaara, and J. Parkkinen. An SVM Classification of Tree Species Radiometric Signatures Based on the Leica ADS40 Sensor. *IEEE Transactions on Geoscience and Remote Sensing*, 49(11):4539–4551, 2011.

[22] J. Heinzel and B. Koch. Investigating multiple data sources for tree species classification in temperate forest and use for single tree delineation. *International Journal of Applied Earth Observation and Geoinformation*, 18:101–110, 2012.

[23] C. M. Kelly, J. S. Wilson, E. A. Baker, D. K. Miller, and M. Schootman. Using google street view to audit the built environment: inter-rater reliability results. *Annals of Behavioral Medicine*, 45(1):108–112, 2013.

[24] I. Korpela, V. Heikkinen, E. Honkavaara, F. Rohrbach, and T. Tokola. Variation and directional anisotropy of reflectance at the crown scale – implications for tree species classification in digital aerial images. *Remote Sensing of Environment*, 115:2062–2074, 2011.

[25] N. Kumar, , P. Belhumeur, A. Biswas, D. Jacobs, J. Kress, I. Lopez, and J. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *European Conference on Computer Vision*, pages 502–516, 2012.

[26] F. Lafarge, G. Gimel'Farb, and X. Descombes. Geometric Feature Extraction by a Multi-Marked Point Process. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1597–1609, 2010.

[27] D. G. Leckie, F. A. Gougeon, S. Tinis, T. Nelson, C. N. Burnett, and D. Paradine. Automated tree recognition in old growth conifer stands with high resolution digital imagery. *Remote Sensing of Environment*, 94:311–326, 2005.

[28] D. Leung and S. Newsam. Proximate sensing: Inferring what-is-where from georeferenced photo collections. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2955–2962, 2010.

[29] Y. Li, D. J. Crandall, and D. P. Huttenlocher. Landmark classification in large-scale image collections. In *Computer*

vision, 2009 IEEE 12th international conference on, pages 1957–1964. IEEE, 2009.

[30] Y. Li, N. Snavely, and D. P. Huttenlocher. Location Recognition using Prioritized Feature Matching. In *European Conference on Computer Vision*, 2010.

[31] Y. Li, N. Snavely, D. P. Huttenlocher, and P. Fua. Worldwide Pose Estimation Using 3D Point Clouds. In *European Conference on Computer Vision*, 2012.

[32] T. Lin, Y. Cui, S. Belongie, and J. Hays. Learning deep representations for Ground-to-Aerial geolocalization. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2015.

[33] T.-Y. Lin, S. Belongie, and J. Hays. Cross-View Image Geolocalization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[34] A. L. Majdik, Y. Albers-Schoenberg, and D. Scaramuzza. Mav urban localization from google street view data. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3979–3986. IEEE, 2013.

[35] K. Matzen and N. Snavely. Scene Chronology. In *European Conference on Computer Vision*, 2014.

[36] S. Mouine, I. Yahiaoui, and A. Verrout-Blondet. Combining Leaf Salient Points and Leaf Contour Descriptions for Plant Species Recognition. In *International Conference on Image Analysis and Recognition*, pages 205–214, 2013.

[37] R. Pu and S. Landry. A comparative analysis of high spatial resolution IKONOS and WorldView-2 imagery for mapping urban tree species. *Remote Sensing of Environment*, 124:516–533, 2012.

[38] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[39] K. L. Roth, D. A. Roberts, P. E. Dennison, M. Alonzo, S. H. Peterson, and M. Bland. Differentiating plant species within and across diverse ecosystems with imaging spectroscopy. *Remote Sensing of Environment*, 167:135–151, 2015.

[40] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. Imagenet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, pages 1–42, 2015.

[41] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Computer Vision–ECCV 2006*, pages 1–15. Springer, 2006.

[42] B. Soheilian, N. Paparoditis, and B. Vallet. Detection and 3d reconstruction of traffic signs from multiple view color images. *ISPRS Journal of Photogrammetry and Remote Sensing*, pages 1–20, 2013.

[43] Q. Sun and D. Batra. Submodboxes: Near-optimal search for a set of diverse object proposals. In *Neural Information Processing Systems (NIPS)*, 2015.

[44] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[45] A. Torii, M. Havlena, and T. Pajdla. From google street view to 3d city models. In *Computer vision workshops (ICCV Workshops), 2009 IEEE 12th international conference on*, pages 2188–2195. IEEE, 2009.

[46] J. Wang, M. Korayem, and D. J. Crandall. Observing the natural world with Flickr. In *IEEE International Conference on Computer Vision Workshop*, pages 452–459, 2013.

[47] L. Waser, C. Ginzler, M. Kuechler, E. Baltsavias, and L. Hurni. Semi-automatic classification of tree species in different forest ecosystems by spectral and geometric variables derived from Airborne Digital Sensor (ADS40) and RC30 data. *Remote Sensing of Environment*, 115:76–85, 2011.

[48] L. Yang, X. Wu, E. Praun, and X. Ma. Tree detection from aerial imagery. In *ACM GIS'09*, 2009.

[49] W. Yao, P. Krzystek, and M. Heurich. Tree species classification and estimation of stem volume and DBH based on single tree extraction by exploiting airborne full-waveform LiDAR data. *Remote Sensing of Environment*, 123:368–380, 2012.