DesPat

Smartphone-Based Object Detection for Citizen Science and Urban Surveys

Getschmann, Christopher; Echtler, Florian

Published in:
i-com

DOI (link to publication from Publisher):
10.1515/icom-2021-0012

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

Research Article

Christopher Getschmann and Florian Echtler*

DesPat: Smartphone-Based Object Detection for Citizen Science and Urban Surveys

https://doi.org/10.1515/icom-2021-0012

Abstract: Data acquisition is a central task in research and one of the largest opportunities for citizen science. Especially in urban surveys investigating traffic and people flows, extensive manual labor is required, occasionally augmented by smartphones. We present DesPat, an app designed to turn a wide range of low-cost Android phones into a privacy-respecting camera-based pedestrian tracking tool to automatize data collection. This data can then be used to analyze pedestrian traffic patterns in general, and identify crowd hotspots and bottlenecks, which are particularly relevant in light of the recent COVID-19 pandemic. All image analysis is done locally on the device through a convolutional neural network, thereby avoiding any privacy concerns or legal issues regarding video surveillance. We show example heatmap visualizations from deployments of our prototype in urban areas and compare performance data for a variety of phones to discuss suitability of on-device object detection for our usecase of pedestrian data collection.

Keywords: citizen science, object detection, pedestrian tracking, smartphone

Figure 1: Smartphone capturing a pedestrian traffic dataset on location.

1 Introduction

Access to smartphones with their multitude of sensors has enabled a wide range of citizen science applications and contributions in the last decade. Some citizen science apps are used as a frontend for manual data input while optionally including built-in sensors, some rely primarily on data provided by these sensors. From marking locations of spotted birds [48] or recording soundbites of nightingale populations [52] to repurposing cameras as air pollution detectors [46], all available sensors are utilized in some way by interested researchers.

When analyzing cities and urban space, evaluating development projects, or backing civic action with quantifiable data, the movement patterns of pedestrians and other traffic participants are a key element. Usually, this data is gathered only by expensive, permanently installed systems or through laborious fieldwork and manual counting, and large datasets have consequently been out of reach for most citizens and many researchers. In addition to cost, automated data collection systems based on computer vision do raise questions about consent, privacy, and compliance with local laws. While low-cost smartphones can already be used to collect manual pedestrian or car counts, the device itself can also gather pedestrian data by using local processing power to run object detection networks on the device itself to process camera input. This approach will help lower the entry barrier for citizens to participate in better knowing their own community, as initially discussed by Jacobs in her seminal work [25] and also later elaborated on by Foth et al. regarding empowering citizens in the digital age [14].

We present an object detection app for mobile devices that allows automated collection of pedestrian movement data in a simple, inexpensive and privacy-preserving way. Smartphones can be placed on windows and balconies to survey public spaces in view of the camera. The on-device object detection is performed with the Tensorflow mobile framework [1] running three different networks of varying speed and precision for different phones, all pretrained on the Microsoft COCO Dataset [31]. Any detections of known

***Corresponding author: Florian Echtler,** Aalborg University, Aalborg, Denmark, e-mail: floech@cs.aau.dk
Christopher Getschmann, Aalborg University, Aalborg, Denmark, e-mail: cget@cs.aau.dk

objects by the network are postprocessed and transformed to geocoordinates, computed from point correspondences between a map and the scene. These can be selected manually by the user on an averaged image of all captures, thereby preserving all image features necessary for mapping points without storing or retaining any footage of individually identifiable pedestrians. Acquired observation data can be easily exported, shared and analyzed in a visualization tool we provide.

Pedestrian data in general can be acquired at three levels of fidelity: individual trajectories, counting data, and spatial distribution densities. DesPat currently allows acquisition and visualization of distribution data, at the advantage of lower processing requirements for inexpensive or outdated smartphones.

Although a server-based solution with far higher processing power would improve the performance allowing trajectory detection, our approach has the advantages of being entirely self-contained, especially regarding network connectivity, and will never collect any sensitive image data which might later be abused in unforeseen ways. In addition, a purely app-based approach allows the creation of datasets with a minimum of hardware cost and technical expertise, thereby enabling large-scale data collection to support novel research avenues.

Our contribution is two-fold: we analyze the feasibility of running object detection networks on inexpensive and low-end mobile devices, and provide an open-source mobile object detector app as a proof of concept for concerned citizens and researchers to collect datasets of pedestrian, cyclist and car densities with very low cost and effort.

2 Related Work

2.1 Citizen Science

Citizen scientist contributions have enabled a wide range of research, either by contributing work, resources, or data. However, two of the major challenges of citizen science are data collection and engaging with the community. Citizens' smartphones can help tackle these issues, so researchers have been relying on them extensively.

The eBird [48] project moved from purely web-based input to a smartphone app that allows citizen scientists and birders to log locations of bird observations. Other projects such as the Naturblick Nightingale app [52], logging nightingale encounters, incorporate additional sensor data and interaction, in this case microphone data. The "Loss of the Night" app [27] relies on the camera, using the image data as the main sensor to measure light pollution. For every built-in sensor, there is a project that makes use of direct or indirect measurements to collect data for their specific research (see Table 1). This range of sensors can even be extended by providing additional sensors or repurposing existing sensors with additional hardware, as demonstrated by the iSpex system [46] which turns the smartphone camera into a spectrometer.

Table 1: Citizen science apps and sensor usage.

Project/App	Sensor
eBird [48]	Manual Data Input
Nightingale [52]	Microphone
Loss of the Night [27]	Camera
iSpex [46]	Camera + external hardware
Cycletracks [42]	GPS
MyShake [26]	Accelerometer

A related approach named Zensors [28] makes use of low-cost smartphones as computer vision sensors, answering questions stated in natural language about image contents. The image data is sent to a server for manual processing by humans first. After receiving a sufficient amount of answers from humans, a machine learning algorithm is trained on this data to take over. Zensors is well suited for tasks which are not clearly defined or very diverse in their requirements.

2.2 Pedestrian Detection

However, for the very specific task of measuring pedestrian traffic automatically, there are specialized hardware and software products. To acquire an overview it is useful to have a look at the most frequently used and installed technology (for a tabular overview see Table 2).

Short-range sensors such as pressure mats and infrared barriers [9] allow accurate measurements for low crowd densities on narrow walkways. While these are relatively inexpensive and therefore widely used sensors, they are unsuited for larger spaces. When privacy is a concern, automated pedestrian counting for open spaces is often conducted with far-infrared/thermal cameras, which allow better foreground/background separation and improve detection accuracy [20]. However, like other commercial systems in this application space, they are prohibitively expensive for citizen scientists. Moreover, most FIR-based counters are limited to a few meters in their operational range. The same is true for stereoscopic depth cameras [44], mostly used in small spaces such as entrances in retail environments.

Table 2: Overview of available automated pedestrian data acquisition solutions.

Range	Sensor	Example	Fixed Location	Accuracy
Short	Infrared Beam	EcoPost [9]	Yes	High
Short	Thermal Camera	Goubet et al. [20]	Yes	High
Short	Stereo Camera	Sensource [44]	Yes	High
Medium	WiFi/Bluetooth-Tracking	Bluemark [2]	Yes	Low
Wide	Cell Data	MotionLogic [36]	No	Low
Wide	Fitness Tracker	Strava Metro [47]	No	Low
Wide	Citizen Science/Personal Data Donations	Cycletracks [42]	No	Medium
Wide	Monocular Camera	Placemeter [38]	No	Medium/High

The usage of smartphone presence detection for counting and tracking passersby, either active or passive, has emerged shortly after the widespread adoption of WiFi-enabled phones. For an extensive survey of this approach, see [8]. Wide-range passive tracking of pedestrians happens by installing static devices in retail stores or public spaces that record WiFi or Bluetooth signals emitted from peoples' phones for commercial [2] or academic purposes [50]. Coarse wide-area position data is also acquired by mobile network operators that track users based on cell tower antenna data and sell access to these pseudonymized datasets. An example is Motionlogic [36], a subsidiary of T-Mobile. Active tracking of smartphone users happens via installed apps or smart fitness trackers. An example for a commercial product is Strava Metro [47], the dataset and visualization product for urban planners by the fitness band company Strava. However, these datasets incorporate mostly sports activities such as jogging or cycling and are primarily useful for analyzing these recreational traffic patterns rather than general traffic. In addition, this data is biased towards social groups that use fitness tracking devices to quantify their workout. Notably, in early 2018, Strava Heatmap data inadvertently exposed previously unknown military bases in the Middle East, thereby highlighting the privacy risks associated with pedestrian traffic data [22]. An example of a volunteer-driven approach is the Cycletracks app in San Francisco [42] which enabled cyclists to share their rides with the San Francisco Transportation Authority. This data was used to help traffic planners understand which impact traffic elements like separated bike lanes have. After the project concluded in 2010, the app has been adapted by 18 other cities at the time of writing.

All these methods have in common that they only provide a rough estimate of pedestrian or cyclist traffic since not every person carries an active smartphone or has the required app installed. Access to datasets from carriers is expensive but offers a higher amount of data in relation to tracked area and time range. In both cases, accuracy is low to very low (on the order of 10–100 m due to wireless cell size), thereby only allowing very generalized statements about pedestrian flows.

Mobile optical systems are the most relevant traffic measurement systems for high-accuracy detection on large-scale spaces. The Miovision Scout [33] is a mobile traffic counter camera. It can be mounted on an extendable pole to survey streets. Computation is done after uploading recorded footage to a server. The modcam [34] is marketed as a pedestrian counter for retail and facility management. It is a monocular ceiling-mounted camera with a fisheye lens running the detection algorithm directly on the device. The startup Placemeter [38] used low-cost Android smartphones to gather image data. Video data from the devices is streamed to a central server where object detection is performed. Customers are billed per video stream and object type (car, pedestrian, etc.). Every phone requires a constant broadband network connection to stream video data and is not supposed to run on battery. Placemeter was acquired by Netgear and the service is no longer available to new customers.

A proof of concept for an open-source mobile object counter is the opendatacam [35]. A Nvidia Jetson Board is used to capture webcam input on-site while processing the video feed with the YOLO object detection algorithm [39]. While the opendatacam approach partly aligns with our goals, there are three major issues. *Cost:* the Jetson board including additional hardware is more expensive than a state-of-the-art Android phone. This makes it infeasible for short-term deployments, citizen science projects, or installations requiring multiple viewing angles. *Safety:* the wiring and lithium-polymer battery connectors need to be assembled by the user and soldered. *Complexity:* the enclosure is complex to build and the board requires soldering skills; setting up the software also requires at least a basic understanding of software development.

Similarly, the more recent Telraam project [49] as part of the WeCount [53] initiative uses a Raspberry Pi for data acquisition and processing. This approach requires less

hardware setup effort than opendatacam, but still is targeted towards tech-savvy users. In contrast, DesPat aims to reduce the required setup to simply installing an app from the official store.

2.3 Visualization and Use of Pedestrian Data

Capturing data is only half of the work, visualizing these datasets is important to infer information. Most pedestrian data visualizations are manually edited maps enriched with manually obtained data, such as maps from Gehl's "Public Life – Public Space" surveys [15] (see Figure 2a).

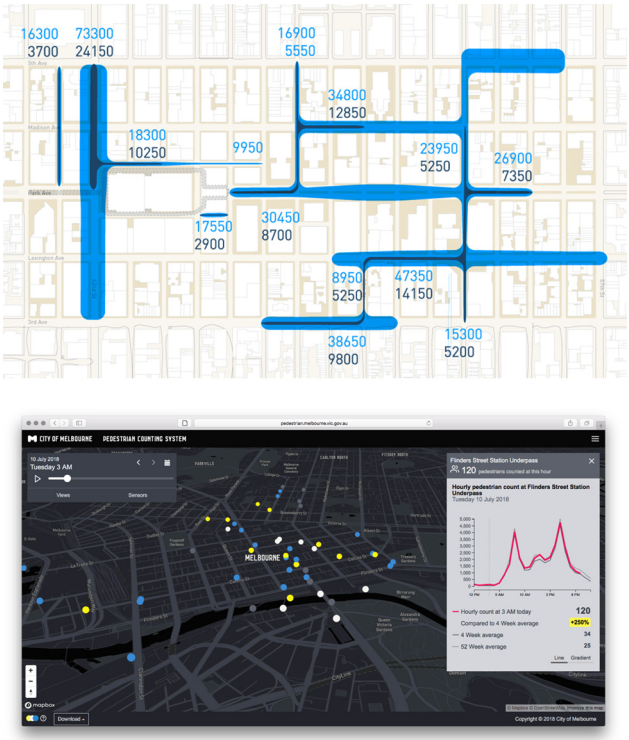

Figure 2: Manual visualization of pedestrian counts in New York City and online visualization for Melbourne.

The City of Melbourne installed permanent pedestrian counters in the city center in 2009 and publishes this information as a part of their open data portal [37]. This is one of few examples where non-manual pedestrian data is visualized (see Figure 2b).

Once data about pedestrian flows is available, it can then be used to analyze the behavior of passersby in everyday situations, but also in relation to short-term installations such as large public displays [13, 41]. Inside larger buildings, pedestrian data can also be collected and used to support Post-Occupancy Evaluation (POE), which would otherwise require large-scale sensor installations such as in [11, 51].

2.4 Summary

Pedestrian tracking based on wireless data offers low-fidelity data for very large areas. Thermal and stereo cameras allow more precise counting in small areas, but are not applicable for larger spaces such as public places. Monocular camera systems such as Placemeter or Miovision Scout make tracking objects on larger public spaces possible, although they are expensive and mostly tied to centralized servers.

Consequently, none of the above-mentioned research or products is aligned with our approach of offering an open dataset acquisition tool suitable for citizen science and urban research. None with the exception of opendatacam or the Cycletracks app is open-source or in any way suitable for low-cost data acquisition or citizen science. While Placemeter did follow the same low-cost approach in terms of raw data acquisition, privacy was not a concern in the architecture of their system. opendatacam and Telraam are similar in their purpose and share design decisions with our approach, but have a far higher entry barrier and are also costly.

3 Pedestrian Detection

Before presenting the design of our app, we will first briefly summarize relevant background information regarding vision-based pedestrian tracking and object detection using convolutional neural networks (CNNs). Two different network architectures are relevant:

Among the best-performing generalized object detection networks is Faster Region-CNN [40] (FRCNN), which uses a two-stage approach. The first stage is the Region Proposal Network which generates proposals for image regions that are likely to contain a detectable object. These region proposals are resized and fed into the second stage, a network used solely for object recognition.

Another approach in contrast to FRCNNs two-stage architecture is handling bounding box regression and classification in a single network: Single Shot Detectors (SSD) [32] generate a fixed number of pairs of class score and bounding box coordinates for each image.

The most prominent advantage of FRCNNs is that proposal generation is run on the input image in original dimensions without resizing, which is helpful for detecting small objects. For SSDs, the image needs to be resized to

Figure 3: Averaged images of all four locations in our own dataset.

a fixed width and height prior to running the network. However, SSD-based networks are faster and have a lower memory footprint.

These architectures are combined with different network types, called base networks, contributing the exact details of filter parameters and image operations. In addition, both types of detection networks can be extended by using Feature Pyramids (FPN) [30], efficient representations for image features at different scales. This increases precision for small objects at the cost of speed.

3.1 Pedestrian Datasets

Whenever machine learning algorithms are used, datasets are as relevant as the algorithms themselves. Regarding pedestrian detection, there are datasets which are specifically created to train and evaluate detection algorithms for persons in urban scenes and general datasets for object recognition and detection of multiple classes. An important example for the latter type of dataset is Microsofts *Common Objects in Context* (COCO) [31], a dataset for object detection consisting of ~200,000 images with 1.5 million objects labeled from 80 classes. These include *person*, *bicycle*, *car* and *truck*.

Datasets which have been compiled to specifically detect pedestrians include *Caltech Pedestrian Detection Benchmark* [7], *KITTI Vision Benchmark Suite* [16], *INRIA Person Dataset* [6], *ETHZ datasets* [12], *TUD-Brussels* [54], *CrowdHuman* [45], *Daimler* [10], and *CityPersons* [56]. All of the datasets above have in common that they are designed to facilitate pedestrian detection for autonomous driving (see also [21]).

Consequently, they have several disadvantages for our application: the resolution is relatively low, the pedestrians in these images are large compared to the total image and sparse at ~1 person per image on average, and the images cover only a single city each [56]. The image data is recorded from a dashcam in a moving vehicle and features no elevated or static viewpoints. CityPersons and CrowdHuman feature more images with larger crowds from a more diverse set of cities and are, therefore, an improvement over other automotive pedestrian datasets in terms of data diversity and size, but still suffer from the same perspective issue as every other dashboard-recorded video frame sequence. None of these datasets is therefore sufficiently similar to a static pedestrian detection setup on an elevated viewpoint to be used for training or evaluation.

To the best of our knowledge, there is no comprehensive dataset of urban scenes that features high-resolution images from a static, elevated viewpoint. Hence, we have decided to evaluate the performance of our system on a suitable self-compiled dataset.

3.2 Evaluation Dataset

Our dataset consists of 160 manually annotated images from 4 different scenes (40 per scene), featuring four elevated observation positions with a downward viewing angle of 30 to 60 degrees (see Figure 3). The scenes show (from left to right) a bus station, a busy intersection, a walkway in a public park, and a residential street corner, and are representative for a variety of urban settings. The 2378 person bounding boxes in the evaluation batch of this dataset have an average size of 83x193 pixel and can be considered small in relation to the overall resolution of our images (11 to 20 megapixels). This is relevant regarding the specific type of neural network used for detection. Some architectures (SSDs, see above) require resizing of the whole input image so the original resolution of an object becomes less important, while others (Faster R-CNN, see above) operate on the original image data and benefit from a high-resolution input. In addition to *person*, the dataset also contains labels for the classes *car*, *truck*, *bus* and *bicycle*, although only the class *person* will be used for this evaluation.

Ground truth annotation of this dataset was done manually using *labelImg* [29]. Bounding boxes are covering the whole extent of the object without padding, even

occluded areas. As stated in [55], the learning process benefits from this setting. Therefore, we also follow this approach for labeling our dataset.

Caltech and other pedestrian datasets included the concept of 'ignore regions' such as extremely crowded parts of an image. In these regions, no detections of the algorithm are rated as false positive or false negative. Our evaluation dataset contains these annotations, but they are currently ignored. Thus, performance is slightly underreported. The size of our dataset is small with 3495 bounding boxes in total (2378 in the class person). This limits the ability to generalize from the evaluation set, but gives a better estimation of the overall performance for our specific use case.

3.3 Object Detection on Mobile Devices

To select the best-suited architecture for our detector on a mobile system, several requirements need to be considered:

Precision and **speed** are always a tradeoff. For the task of recording the paths of pedestrians precision is a higher priority than speed, for counting pedestrians it is vice versa. For both modes, faster networks reduce battery consumption and increase the total observation time.

Memory: on many smartphones, processing power is not the limiting factor, but rather memory capacity and speed. A big network may perform well on modern phones with 2–4 GB RAM but will run out of memory on older models.

Object size. Pedestrians are small objects in high-resolution images of public spaces. The network needs to detect objects at a lower scale reliably. This is our main problem which needs to be addressed.

To test different networks on Android we use the Tensorflow Mobile Framework [17] in combination with the Tensorflow Object Detection API [18]. This allows us to use the networks in the Tensorflow model zoo which have been pre-trained on the COCO dataset [31].

We compare several SSD and FRCNN models with different base recognition networks (see Figure 4). The left bar (blue) represents average precision as mAP for the class *person* (higher is better). The right bar (green) represents the complete processing time of the network from image input to object output for one image. Runtime was measured on a mid-level smartphone (LG G6). The absolute runtime is device-dependent, but the relative runtime will remain fairly constant.

In terms of speed, Mobilenets in both versions [23, 43] as a base for SSD outperform the larger FRCNN networks.

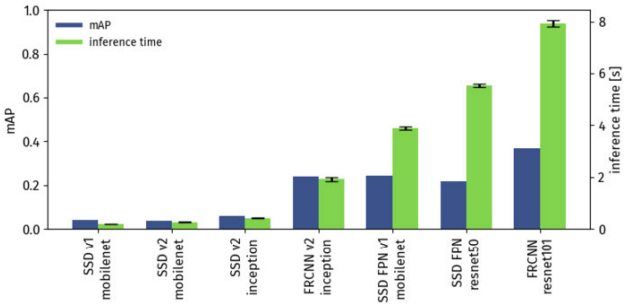

Figure 4: Comparison of detection algorithms on full image (higher is better for precision, lower is better for runtime).

The lower precision of SSD-based models compared to FRCNN networks can be explained by their fixed input size. The SSD FPN models require rescaling to 640 pixels, all other SSD models to 300 pixels before processing. FRCNN networks deliver better precision, but also have higher memory consumption and are slightly slower.

To further improve performance for the task of detecting small pedestrians in high-resolution images, several points must be taken into consideration.

3.4 Image Tiling

One of the most promising approaches of *upscaling* the whole image as proposed for the Citypersons dataset [56] is not an option on a mobile platform, as this increases memory consumption and is only suitable for FRCNN networks. However, an effect similar to upscaling without the same memory requirements can be achieved by a sliding-window approach. Consequently, we split the full-resolution input image in non-overlapping tiles of constant size, downscale each tile to the network's fixed input size, and detect objects in every tile separately.

When applying this approach to the best-performing networks in terms of speed and precision, both computation time and precision improve considerably (see Figure 5). The mobilenet-based SSD has a fixed input size of 300 pixels and shows the best tradeoff between accuracy and speed at image tiles of about twice its input size. If Feature Pyramids are used (at 640 pixels input size), the image tiles can be increased to three times the input size before the models begin to show a strong decrease in precision. This shows that architectures with Feature Pyramids have an advantage for detecting small objects in our evaluation dataset, while differences between the base networks (resnet50 or mobilenets) can be neglected for the SSD FPN architecture.

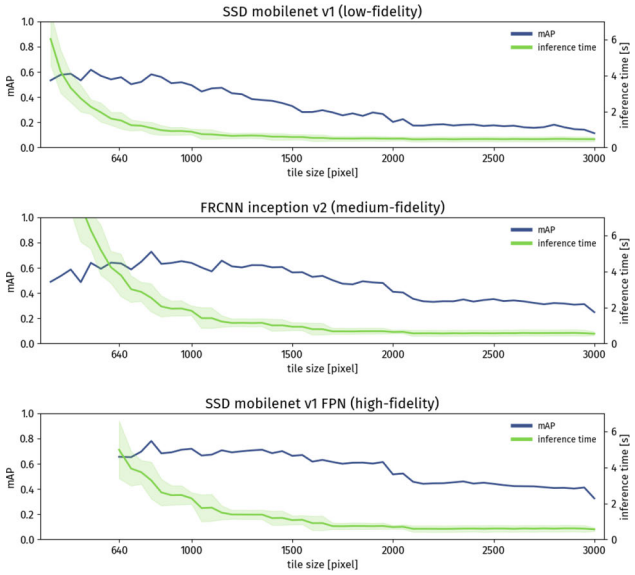

Figure 5: Precision and runtime for a full image in relation to tile sizes. Inference time was measured across the whole evaluation set on a GPU (Nvidia GeForce GTX1050Ti). The FPN model has a larger input size and is evaluated starting at 640 pixels. X-axis increments: 50 pixels.

When splitting the image into tiles, two issues are introduced. There are detection *artifacts* when objects are split across tile boundaries (causing either no detection or double detection) and a *reduced field of view* as fixed tile sizes which are not an exact divisor of the image size will exclude areas around the image border, resulting in false negatives.

3.5 Transfer Learning

We assume that the precision of the networks, pre-trained on COCOs 80 classes, can be further improved by transfer learning and fine-tuning on a more specific dataset. The main difference between COCO's examples for the *person* class and our dataset are object sizes and viewing angles.

To evaluate possible gains achieved by transfer learning, we split our evaluation dataset into 120 images (3 scenes) for training and 40 images (1 scene) for evaluation. With a subset of three COCO-classes (person, bicycle, car) we retrained the fastest but worst-performing network SSD mobilenet v1 on 5490 randomly-cropped image tiles derived from the training set images. However, when evaluating the precision of fine-tuned and original network on the remaining 40 images, no significant improvement could be reported. We assume that examples of persons from our dataset and COCO are too close for transfer learning to yield any substantial precision gains when trained on our small evaluation dataset. Expanding our dataset and reevaluating transfer-learning is part of the future work.

3.6 Results

While FRCNN outperforms SSD variants in terms of precision, even small FRCNN networks have a higher memory consumption than every SSD model. When choosing SSD models, we recognize that feature-pyramid networks (FPN) have a clear advantage over standard variants. The two FPN networks, SSD with mobilenets v1 and resnet50 as base networks perform at the same level while the resnet network has a slightly higher inference time and binary size. While FPN SSDs perform well on larger tiles than regular SSDs, their inference time is considerably higher, especially on mobile devices. Consequently, we settle on the SSD mobilenet v1 network as the default object detection network for our app. We use non-overlapping tiles to improve the precision at small scales for SSD networks. Since this network offers the highest speed at the expense of precision, we call this the Low-Fidelity option. Phones with sufficient computing power can optionally use the FRCNN inception v2 model (Mid-Fidelity) or the SSD mobilenet v1 FPN network (High-Fidelity) for increased precision.

4 App & Visualization

4.1 DesPat Android App

We selected Android as our target platform, as we want to make use of inexpensive smartphones and need to access the camera without user interaction. Our app is targeted at phones running Android 6.0 (API level 23) and above. At the time of writing, this corresponds to about 74.8 percent of all active devices using the Google Play Store [19].

The core functions of our app consist of five distinct parts:

Image Capturing. For periodic image capture, the app needs to trigger the shutter at fixed intervals without any user interaction (and not when the system schedules it). The capturing service can run in two different modes: *persistent* and *non-persistent camera*.

In non-persistent mode, the camera module is closed after each capture. It is powered down and an event is scheduled using Androids Alarm Manager to wake up in time for the next capture. Once this happens, the camera is initialized and a full metering run is started to get a focus lock as well as determine exposure and white balance. This may take up to 1.5 seconds, depending on the illumi-

nation and hardware. By allowing the device to sleep in between captures, more than a full day of battery life can be achieved. The minimum shutter interval in this mode is 6 seconds.

In persistent mode, the camera is initialized at startup and runs permanently in preview mode. Consequently, both the processor as well as the camera module are powered up and active. When the camera is triggered, images can be captured without delay as the camera is active and autofocus/auto-exposure have already been finished. This mode increases power consumption and heats up the device, but allows shutter intervals of less than 6 seconds. 3 hours of battery life are common in this mode among tested phones if running at maximum possible speed, and about 10 to 12 hours at one image per minute.

Detection. After image capture the footage is analyzed for pedestrians and other tracked objects. Processing of the buffered images is done in batches by the detector service, which runs once a minute when the phone wakes up and takes care of the costly process of running the detection algorithm on every image. After the algorithm has processed the image, no image data is retained. The detection algorithms are discussed in detail in Section 3.

Homography Transformation. The detection algorithm returns bounding boxes for detected objects, which need to be transformed to map coordinates. Before transformation, bounding boxes are reduced to a single point. For cars, this is the center point of the bounding box; for pedestrians and bicycles, the center of the lower bounding box boundary (e. g. a person's feet). To transform these image points to map coordinates, a homography transformation is calculated. At least four corresponding points are required, which have to be manually selected on a map and an image of the scene (Figure 6).

Image Averaging. The corresponding points to compute the homography matrix need to be selected manually, however, doing that on location is usually not feasible. Either the phone is inaccessible, or touching the phone while mounted may change the camera orientation, thereby rendering the point correspondences invalid. The alternative is storing an image and selecting the correspondences after capturing. To avoid inadvertent capture of unsuspecting pedestrians, we average all captured images to create a mean image containing only static objects. Even persons standing or sitting in place will blur after processing only a few images. To achieve a fully pedestrian-less scene, a minimum of 30 to 50 images is usually sufficient, depending on the amount of movement in the scene and the pixel size of the non-static objects.

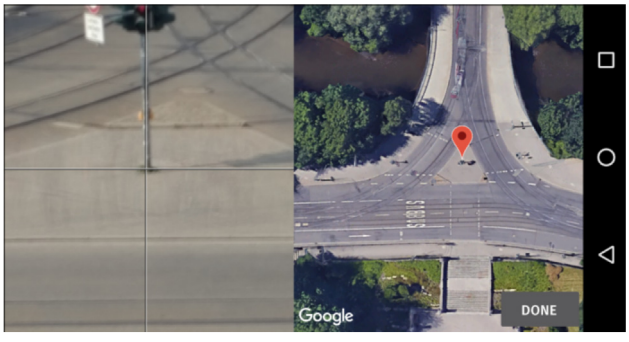

Figure 6: User interface for selecting image to map point correspondences.

Data Export. All data from a single capturing session can be exported as a ZIP archive by using the Android share functionality. This allows to save the file on the device, in a shared cloud folder, or directly attach it to an email. Each archive contains three files: a CSV file with all *detections*, including confidence values, class, and coordinates; a JSON file with *session metadata* such as device name and homography points; and the *averaged result* from all captured images as JPEG.

4.2 VizPat Visualization Tool

The exported CSV data can be processed with spreadsheet software such as Excel and visualized with Google Maps, but neither of these tools performs well on the task of visualizing pedestrian density data. Since data acquisition is pointless without means of examining and evaluating, we present our own visualization tool:

Density data for arbitrary object classes is displayed as a binned heatmap or in a raw scatter plot on top of a map layer, which can be loaded from Google Maps or OpenStreetMap. All layers are interactive and can be turned on or off. The map allows panning and zooming; the heatmap parameters regarding bin size and opacity can also be changed. Data can be filtered by time, object class and recording device. Datasets gathered in parallel from different phones or sequentially from the same device can be combined. The JavaScript library D3 [3] is used to interactively manipulate the visualization and to filter data points.

We choose a heatmap with hexagonal bins as default visualization [4], since binned map positions are easy to visually parse and do not give a false presumption of location accuracy in contrast to a scatterplot (which is also available as an optional visualization). The hexagon is an obvious choice for tile shape since its geometry is efficient to compute and it is the highest-order polygon that allows tiling without overlapping.

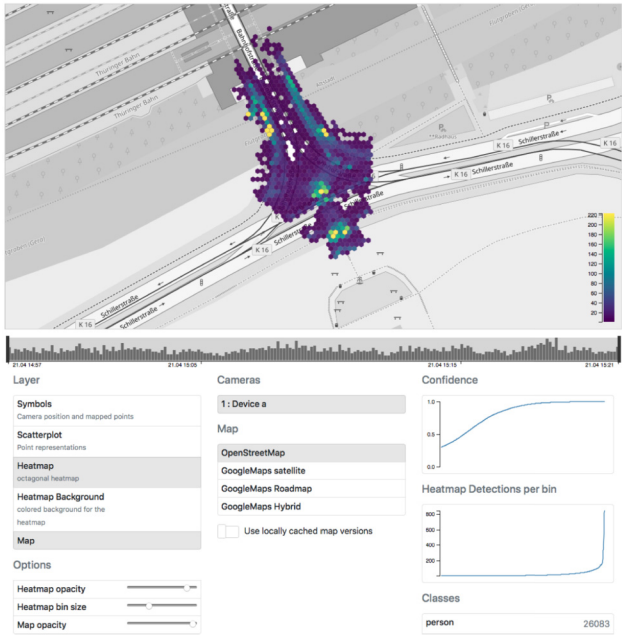

Figure 7: Web-based visualization tool with heat map (top), timeline (center), and user settings (bottom).

For a live demo, see http://despat.de/visualize/.

5 Performance Evaluation

Our evaluation will focus on precision and speed of the detector network. *Precision* is reported as Mean Average Precision (mAP), calculated similarly to the COCO evaluation as the area under the full interpolated precision-recall-curve (but only with an Intersection over Union threshold of 0.5 because the improved localization metric using multiple thresholds is not required). For a detailed explanation of the mAP calculation, see [24]. All *speed* measurements are given per image, independent of camera resolution. When evaluating inference time measurements it should be noted that all computation was done on the CPU of the smartphone as no GPU support for all used network models was available at the time of writing. This may change in the near future and we assume that some phones in the mid- to high-cost section will improve their performance with the Neural Networks Interface introduced in Android 8.0.

5.1 Performance of the Detector

We evaluate three networks: SSD mobilenet v1 (low-fi), FRCNN inception v2 (mid-fi), and SSD mobilenet v1 FPN (high-fi), running on limited hardware in terms of processing power, memory and battery. Our reference network is the best-performing available network from the Tensorflow model zoo, trained on COCO and executed on a desktop GPU (NVidia GeForce GTX1050Ti). Ground truth was compiled by a human without time constraints and with access to the whole image and the complete image series.

Table 3: Precision of the three networks used in the app (mAP [24] for class *person*). The reference network always uses 6 tiles per image and does not suffer from the tiling error of false negatives at image borders.

Network	mAP
SSD mobilenet v1 @ 800 px (low-fi)	0.54
FRCNN inception v2 @ 800 px (mid-fi)	0.65
SSD mobilenet v1 FPN @ 800 px (high-fi)	0.78
FRCNN NAS @ 1/6 (reference)	0.86
Ground Truth	1.00

The low-fi network only achieves a relatively low overall mAP of 0.54. However, the mid-fi and high-fi networks achieve an average precision of 0.65 and 0.78 respectively, which compares very well to the reference network with an mAP of 0.86. An average precision of 0.78 and above on our very challenging dataset, without any additional postprocessing, can be noted positively. In addition, the reference network runs about 30 times slower than our high-fi network when both are executed on the same desktop GPU.

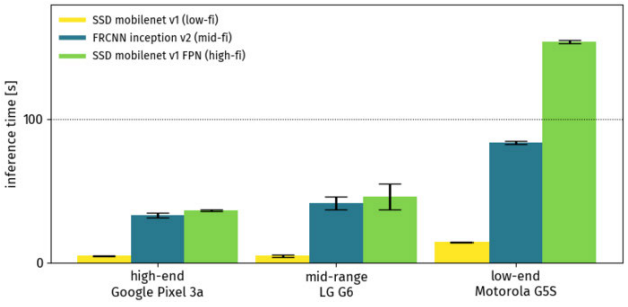

Figure 8: Runtime of the networks on different devices at native camera resolution, optimized for maximum precision (error bar is standard deviation).

The phones used as test cases have been chosen from all price ranges over the last 2 years. The Google Pixel 3a is priced around 450 USD, representing the higher-end device class (by performance). Mid-range phones are represented by the LG G6, available at about 230 USD, while low-end phones such as the Motorola 5GS are available for less than 130 USD.

Results from the *low-fidelity network* highlight the runtime-precision tradeoff mentioned above. It runs at just 4.5 s on the high-end and 4.6 s on the mid-range phone, thereby easily allowing a capture rate of 5 to 6 images per minute. The performance on the low-end device is worst at 14.3 s.

The *mid-fidelity network's* runtime is 32.7 s and 41.5 s on high and mid-range phones. The low-end phone finishes processing after 83 s, setting the maximum capture interval to about 0.5 images per minute.

The *high-fidelity network* takes only marginally more time on the high and mid-level models, running for 36.4 and 45.9 s. Although it is possible to run the high-fidelity network on the low-end phone, it is prohibitively slow at about 153 s.

While some phones will be slower in detection, their maximum in precision is potentially higher due to their advantage in resolution. However, a gain in resolution cannot be directly translated to gains in precision due to inexpensive optical systems, small sensor size, and resulting poor image quality.

Note that all these measurements were conducted with networks running at maximum resolution and the smallest feasible tile size and thus are the upper boundary. If a phone is placed in a store entrance or on a balcony surveying only a narrow street, distance from the camera to pedestrians is low and their size in the image is considerably higher. This allows to increase the tile size and reduces the number of overall tiles, thereby improving runtime.

In conclusion, we currently select the low-fi network as the default choice, even though it has relatively low precision. However, its runtime stays below 10 s on most devices, even at full resolution, thereby at least partially compensating for missed detections through a higher update rate.

5.2 Performance and Thermal Limits

The heat management capacities of smartphones are not designed to support long periods of full CPU load. When running a phone constantly at full load, the temperature causes processor throttling, which increases processing time. This sets device temperature as the upper boundary of performance rather than computing resources of the processor. When running the camera subsystem permanently (camera is active and providing a preview datastream), battery temperature increases by 10 °C above ambient. We determined empirically that about 50 to 70 % CPU utilization keeps most phones within safe working battery temperatures of 50 °C or less. Consequently, if the individual combination of phone and network requires 5 seconds per image, a recommended shutter interval of 10 seconds is suggested to the user.

6 Example Results

Based on our performance analysis, we conclude that it is indeed feasible to support citizen science through object detectors on inexpensive mobile devices. We present two real-world examples of recorded datasets and discuss these results (running the high-fi network).

The first dataset in Figure 9 from the rear entrance of the train station in Erfurt, Germany is a good example of the amount of space a single camera can survey when placed well. It is clearly visible which routes people prefer to cross the street. When the visualization tool is used, it is even possible to see the arrival time of trams and trains on the timeline. Live visualization for this dataset is available at http://despat.de/visualize/#dataset_erfurt.

The second dataset in Figure 10 was recorded at a the Summaery festival in Weimar, Germany by four devices simultaneously. When using multiple phones in parallel, a more complex geometry of a public space (or simply larger spaces) can be surveyed. However, the visualization tool will currently naively merge overlapping areas, pedestrians that have been picked up by several cameras at once are simply added up. Live visualization for this dataset is available at http://despat.de/visualize/#dataset_summaery.

7 Discussion

Based on experiences from our deployments for urban survey datasets described above, we conclude that a downward-facing viewing angle of approximately 30 degrees is best-suited for urban spaces. At lower angles, pedestrians will start occluding each other in groups, while at higher angles, the raw images will differ significantly from the person images the neural network was trained with, thereby lowering detection rate. In our station dataset, the distance between the camera and the farthest detections is approximately 80 m, while the closest detections are about 14 m from the camera. This is caused by image tiling: at lower distances, pedestrians will be split across two or more image tiles, thereby inhibiting detection. This problem can be mitigated by using larger tile

Figure 9: Train station rear entrance, Erfurt (top: averaged camera view, bottom: map visualization). Map image data: ©Google 2018, GeoBasis-DE/BKG ©2009.

sizes if close-range detection is desired, e. g. in smaller indoor spaces where the minimum distance is likely to be lower. Despite these limitations, our app is able to provide data on urban movement patterns with far less staff resources than common methods – a high-end device is potentially able to run for over 12 hours continuously, which could only be achieved with several persons taking turns during a manual survey and would already constitute an Accuracy Level 3 survey according to the guidelines of the City of London [5].

During our deployments, we noted that while our processing workflow is designed to take people's privacy into account and does not retain images, there is no way to communicate this without additional measures. When running unattended setups from elevated viewpoints, it may be advisable to keep the smartphone unobtrusive. In all other cases, one needs to be prepared to engage with pedestrians and discuss the work, however, this was not an issue at any deployment as people tend to be curious and are generally pleasant.

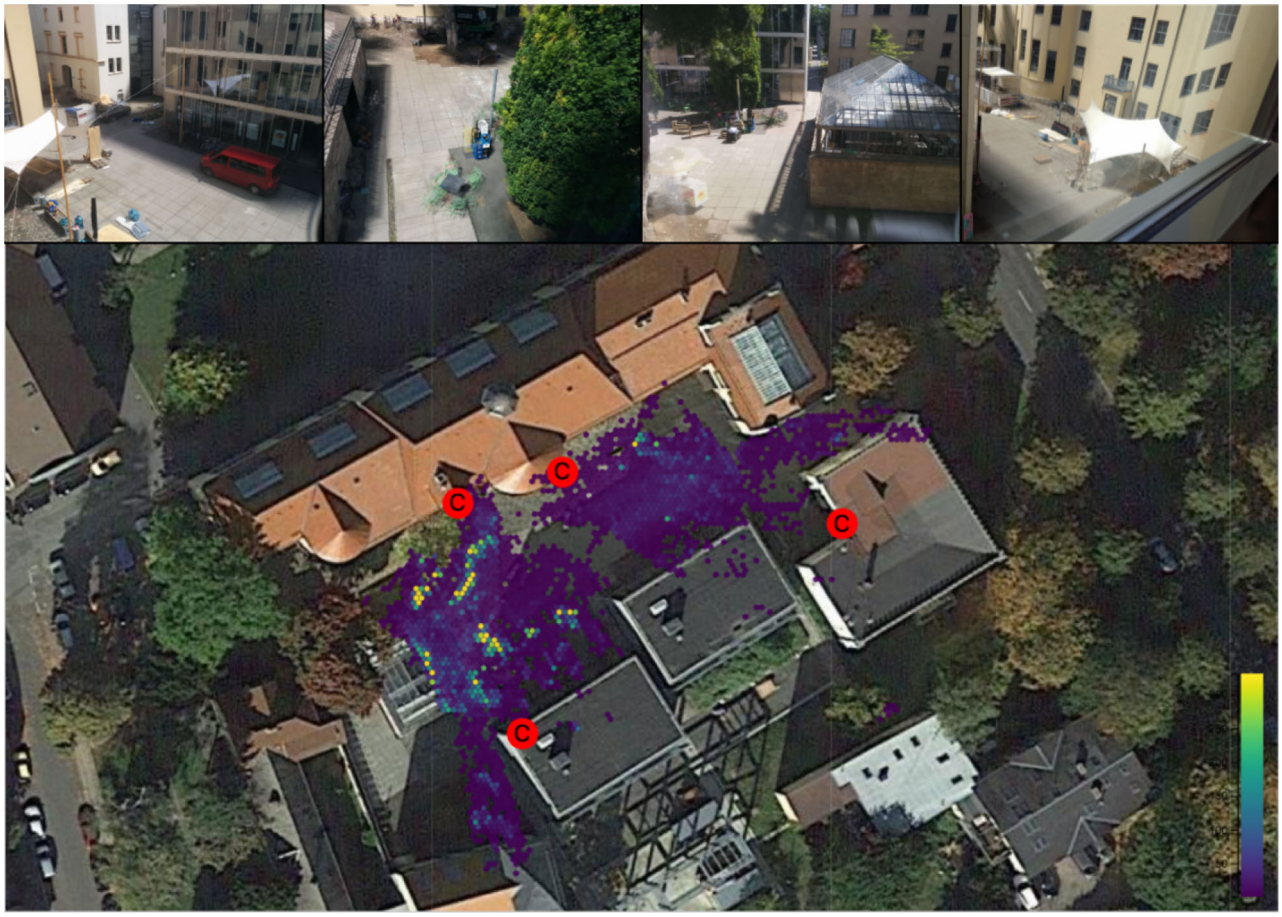

Figure 10: Summaery festival, Weimar (top: averaged camera views, bottom: map visualization with camera locations). Map image data: ©Google 2018, GeoBasis-DE/BKG ©2009.

The collected data is currently does not provide additional attributes of passersby, such as group size, age, or walking speed, which can arguably be collected through manual surveys. We discuss potential extensions to alleviate some of these limitations in the future work.

The approach of tiling images to increase detection precision for tiny objects is suitable for the specific problem at hand. While this is a tradeoff between speed and precision, we conclude that this is the most feasible option. Especially for small objects, the temporal resolution still is limited, allowing us to record occurrences at fixed intervals but not the individual paths. This yields neither trajectory data nor counting data, but allows object identification (*Is this a pedestrian, a cyclist or a car?*). While tracking individual trajectories of objects would be possible using a different computer vision approach, tracking trajectories and object classes of small objects at the same is not feasible given commodity smartphones. However, given the rise in image processing capabilities of modern smartphones, this compromise may be outdated in a few years' notice.

Nevertheless, we are already able to evaluate general traffic patterns and desire paths (*Where do people cross a public space?*), quantify dwelling time (*The bikelane in front of my house is blocked by illegally parked cars for 6 hours per day on average.*), give relative usage estimation (*Pedestrian density at 9 am is twice as high as at 11 am.*), or analyze crowd hotspots which should be avoided during the COVID-19 pandemic (*Bus stop is overcrowded between 7 and 8 am.*).

While our app is compatible with every device running Android Version 6.0 or later, detector performance relies heavily on the hardware of the device. By offering faster but less precise networks and using less tiles per image, we can still utilize even very slow hardware. However, precision also relies on the camera system and especially budget hardware with 5 or 8 megapixel cameras are not suitable for large spaces. Nevertheless, these low-end devices can still be used for small streets and pathways.

8 Future Work

We are currently planning real-world deployments with urban studies and architecture researchers exploring potential integration of our app into their data acquisition procedures. Historic and current data used in this context is overwhelmingly pedestrian counts, sampled in 10 to 60 minute timeslots. To include our app well in already existing data acquisition procedures, the additional functionality to record reliable counts in addition to pedestrian paths is required and is part of our future work. We also will investigate how DesPat can be integrated with related, more centralized projects such as Telraam [49] without compromising the privacy of passersby.

Future workshops nonwithstanding, however, we publish our app, all source code, and additional materials (datasets etc.) under permissive licenses to support adaption and modification of resources.

There are three main categories we want to improve in the app itself: we want to add functionality regarding data features, improve the performance of the detector, and make our app more accessible for usage in a workshop scenario.

Counting and Action Labels: We plan to introduce action labels for detected objects (static and moving) based on inter-image comparison. This allows answering the question where people are moving through, and where people need or want to spend time lingering around. In addition, by generating object paths, the movement speed can be calculated and accurate counting becomes possible. We will investigate if this can be reliably done for small regions of the image (e. g. zebra crossings, street corners, etc) with an increased update rate. Additionally, other object classes besides pedestrians could then be counted, such as fast-moving bicycles or cars.

Extending the dataset and transfer learning: Extending our dataset of ground-truth annotated high-resolution pedestrian images is a priority. We are exploring options how this could be integrated into a workshop format with interested researchers. Once a larger dataset is available, we will reevaluate the precision improvements achievable with transfer-learning for our specific problem domain of small-objects with a downward facing angle.

Ignore Regions: Not all parts of an image may contain pedestrians. When starting a new session, the user could mark ignore regions such as sky or buildings. When the image is subdivided into tiles, all tiles that are covered by an ignore region can be skipped during detection. This would reduce processing time as well as reducing false positives from reflections in mirror facades or people visible through windows.

9 Conclusion

We present an end-to-end open-source solution to enable researchers and citizen scientists alike to collect and analyze pedestrian movement data using object detection algorithms on widely-available smartphones. Our app uses convolutional neural networks running directly on a wide range of Android phones to enable automated data acquisition while avoiding potentially privacy-sensitive image storage or transport. We conclude that it is indeed feasible to augment citizen science apps by object detectors on smartphones of all price-ranges, even old or inexpensive devices. In the case of data acquisition for urban surveys, the tradeoff between runtime and accuracy is suitable for detecting pedestrians in order to evaluate their paths or a general assessment of crowdedness and space allocation in a given timeframe.

Further work will focus on improving detection performance, and exact counting of objects in small areas. We encourage other researchers to evaluate if their research can benefit from this automatized and inexpensive way of data acquisition. All resources (evaluation dataset with annotations, source code, data from live deployments) are available under permissive licenses to reuse and adapt at http://despat.de/visualize/#dataset_summaery.

Reproduction Note

The application, source code including documentation, anonymized data, and scripts for generating all referenced plots is available publicly: https://github.com/volzotan/despat.

Funding: This work was partially supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) through individual grant EC437/1-1.

References

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*. 265–283. https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[2] Bluemark Innovations. 2021. Products. Website. Retrieved Jan 10, 2021 from https://bluemark.io/products/.
[3] Mike Bostock. 2021. Data-Driven Documents (D3). Website. Retrieved Jan 10, 2021 from https://d3js.org/.
[4] D. B. Carr, R. J. Littlefield, W. L. Nicholson, and J. S. Littlefield. 1987. Scatterplot Matrix Techniques for Large N. *J. Amer. Statist. Assoc.* 82, 398 (1987), 424–436. https://doi.org/10.1080/01621459.1987.10478445.
[5] City of London. 2021. Measuring Pedestrian Activity. Website. Retrieved Jan 10, 2021 from https://www.polisnetwork.eu/uploads/Modules/PublicDocuments/london_lip_measuring_pedestrian_activity.pdf.
[6] Navneet Dalal. 2005. INRIA dataset. Website. Retrieved Jan 10, 2021 from http://pascal.inrialpes.fr/data/human/.
[7] Pjotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. 2009. Pedestrian Detection: A Benchmark. In *Proceedings of CVPR 2009*. http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/.
[8] Adriana Draghici and Maarten Van Steen. 2018. A Survey of Techniques for Automatically Sensing the Behavior of a Crowd. *ACM Comput. Surv.* 51, 1, Article 21 (Feb. 2018), 40 pages. https://doi.org/10.1145/3129343.
[9] Eco Compteur. 2021. Pyro Post. Website. Retrieved Jan 10, 2021 from https://www.eco-compteur.com/en/products/pyro-range/urban-post.
[10] Markus Enzweiler and Dariu M. Gavrila. 2009. Monocular Pedestrian Detection: Survey and Experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 12 (Dec. 2009), 2179–2195. https://doi.org/10.1109/TPAMI.2008.260.
[11] Varick L. Erickson, Alex Beltran, Daniel A. Winkler, Niloufar P. Esfahani, John R. Lusby, and Alberto E. Cerpa. 2013. TOSS: Thermal Occupancy Sensing System. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (Roma, Italy) *(BuildSys'13)*. ACM, New York, NY, USA, Article 35, 2 pages. https://doi.org/10.1145/2528282.2534155.
[12] Andreas Ess, Bastian Leibe, Konrad Schindler, and Luc van Gool. 2008. A Mobile Vision System for Robust Multi-Person Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. IEEE Press.
[13] Patrick Tobias Fischer and Eva Hornecker. 2012. Urban HCI: Spatial Aspects in the Design of Shared Encounters for Media Facades. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI'12)*. ACM, New York, NY, USA, 307–316. https://doi.org/10.1145/2207676.2207719.
[14] Marcus Foth, Laura Forlano, Christine Satchell, and Martin Gibbs. 2011. *From Social Butterfly to Engaged Citizen: Urban Informatics, Social Media, Ubiquitous Computing, and Mobile Technology to Support Citizen Engagement*. The MIT Press.
[15] Jan Gehl. 2004. *Public spaces – public life*. Arkitektens Forlag.
[16] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2013. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)* (2013).
[17] Google. 2021. tensorflow Mobile. https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android/.
[18] Google. 2021. tensorflow Object Detection API. https://github.com/tensorflow/models/tree/master/research/object_detection.
[19] Google 2021. Play Store Distribution Dashboard. Website. Retrieved Jan 10, 2021 from https://developer.android.com/about/dashboards/.
[20] Emmanuel Goubet, Joseph Katz, and Fatih Porikli. 2006. Pedestrian tracking using thermal infrared imaging, Bjørn F. Andresen, Gabor F. Fulop, and Paul R. Norton (Eds.). 62062C. https://doi.org/10.1117/12.673132.
[21] Irtiza Hasan, Shengcai Liao, Jinpeng Li, Saad Ullah Akram, and Ling Shao. 2020. Generalizable Pedestrian Detection: The Elephant In The Room. arXiv:2003.08799 [cs.CV].
[22] Alex Hern. 2018. Fitness tracking app Strava gives away location of secret US army bases. Website. Retrieved Jan 10, 2021 from https://www.theguardian.com/world/2018/jan/28/fitness-tracking-app-gives-away-location-of-secret-us-army-bases.
[23] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861 [cs] (April 2017).
[24] Jonathan Hui. 2021. mAP (mean Average Precision) for Object Detection. Website. Retrieved Jan 10, 2021 from https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173.
[25] Jane Jacobs. 1961. *The Death and Life of Great American Cities*. Random House.
[26] Qingkai Kong, Richard M. Allen, and Louis Schreier. 2016. MyShake: Initial observations from a global smartphone seismic network. *Geophysical Research Letters* 43, 18 (2016), 9588–9594. https://doi.org/10.1002/2016GL070955.
[27] Christopher Kyba. 2021. Loss of the night. Website. Retrieved Jan 10, 2021 from http://www.verlustdernacht.de/Loss_of_the_Night_App_engl.html.
[28] Gierad Laput, Walter S. Lasecki, Jason Wiese, Robert Xiao, Jeffrey P. Bigham, and Chris Harrison. 2015. Zensors: Adaptive, Rapidly Deployable, Human-Intelligent Sensor Feeds. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI'15)*. ACM, New York, NY, USA, 1935–1944. https://doi.org/10.1145/2702123.2702416.
[29] Tzu Ta Lin. 2021. labelImg. https://github.com/tzutalin/labelImg/.
[30] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2016. Feature Pyramid Networks for Object Detection. arXiv:1612.03144 [cs] (Dec. 2016).
[31] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. 2014. Microsoft COCO: Common Objects in Context. arXiv:1405.0312 [cs] (May 2014).
[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: Single Shot MultiBox Detector. arXiv:1512.02325 [cs]. 9905 (2016), 21–37. https://doi.org/10.1007/978-3-319-46448-0_2.
[33] Miovision. 2021. Miovision Scout. Website. Retrieved Jan 10, 2021 from http://miovision.com/data-link/scout/.
[34] Modcam. 2021. Modcam. Website. Retrieved Jan 10, 2021 from https://www.modcam.com/.

[35] moovel abs. 2021. opendatacam. Website. Retrieved Jan 10, 2021 from https://opendatacam.moovellab.com/.
[36] MotionLogic. 2021. Understanding Movement Streams. Website. Retrieved Jan 10, 2021 from https://www.motionlogic.de/blog/en/solutions/.
[37] City of Melbourne. 2021. Open data portal. Website. Retrieved Jan 10, 2021 from http://www.pedestrian.melbourne.vic.gov.au/.
[38] Placemeter, Inc. 2020. Placemeter. Website. Retrieved Jul 07, 2020 from https://web.archive.org/web/20200707203740/http://www.placemeter.com/.
[39] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. arXiv:1804.02767 (2018).
[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497 [cs] (June 2015).
[41] Hasibullah Sahibzada, Eva Hornecker, Florian Echtler, and Patrick Tobias Fischer. 2017. Designing Interactive Advertisements for Public Displays. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI'17)*. ACM, New York, NY, USA, 1518–1529. https://doi.org/10.1145/3025453.3025531.
[42] San Francisco County Transportation Authority. 2021. Cycletracks App for iOS and Android. Website. Retrieved Jan 10, 2021 from https://www.sfcta.org/modeling-and-travel-forecasting/cycletracks-iphone-and-android.
[43] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. arXiv:1801.04381 [cs] (Jan. 2018).
[44] Sensource. 2021. Sensource People Counters. Website. Retrieved Jan 10, 2021 from https://www.sensourceinc.com/hardware/people-counters/.
[45] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. 2018. CrowdHuman: A Benchmark for Detecting Human in a Crowd. arXiv:1805.00123 [cs.CV].
[46] Frans Snik, Jeroen H. H. Rietjens, Arnoud Apituley, Hester Volten, Bas Mijling, Antonio Di Noia, Stephanie Heikamp, Ritse C. Heinsbroek, Otto P. Hasekamp, J. Martijn Smit, Jan Vonk, Daphne M. Stam, Gerard van Harten, Jozua de Boer, and Christoph U. Keller. 2014. Mapping atmospheric aerosols with a citizen science network of smartphone spectropolarimeters. *Geophysical Research Letters* 41, 20 (2014), 7351–7358. https://doi.org/10.1002/2014GL061462.
[47] Strava. 2021. Strava Metro. Website. Retrieved Jan 10, 2021 from https://metro.strava.com/.
[48] Brian L. Sullivan, Christopher L. Wood, Marshall J. Iliff, Rick E. Bonney, Daniel Fink, and Steve Kelling. 2009. eBird: A citizen-based bird observation network in the biological sciences. *Biological Conservation* 142, 10 (Oct. 2009), 2282–2292. https://doi.org/10.1016/j.biocon.2009.05.006.
[49] Telraam project partners. 2021. Telraam. Website. Retrieved Jan 08, 2021 from https://www.telraam.net/en/what-is-telraam.
[50] Martin W. Traunmueller, Nicholas Johnson, Awais Malik, and Constantine E. Kontokosta. 2018. Digital footprints: Using WiFi probe and locational data to analyze human mobility trajectories in cities. *Computers, Environment and Urban Systems* 72 (Nov. 2018), 4–12. https://doi.org/10.1016/j.compenvurbsys.2018.07.006.
[51] Himanshu Verma, Hamed S. Alavi, and Denis Lalanne. 2017. Studying Space Use: Bringing HCI Tools to Architectural Projects. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI'17)*. ACM, New York, NY, USA, 3856–3866. https://doi.org/10.1145/3025453.3026055.
[52] Silke Voigt-Heucke. 2021. Naturblick Nightingale. Website. Retrieved Jan 10, 2021 from https://forschungsfallnachtigall.de/.
[53] We-Count project partners. 2021. We-Count. Website. Retrieved Jan 08, 2021 from http://we-count.net/.
[54] Christian Wojek, Stefan Walk, and Bernt Schiele. 2009. Multi-cue onboard pedestrian detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (June 2009), 794–801. https://doi.org/10.1109/CVPR.2009.5206638.
[55] Shanshan Zhang, Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. 2016. How Far are We from Solving Pedestrian Detection? arXiv:1602.01237 [cs] (Feb. 2016).
[56] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. 2017. CityPersons: A Diverse Dataset for Pedestrian Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 4457–4465. https://doi.org/10.1109/CVPR.2017.474.

Bionotes

Christopher Getschmann
Aalborg University, Aalborg, Denmark
cget@cs.aau.dk

Christopher Getschmann is a PhD candidate in the Human-Centered Computing group at Aalborg University. His research focuses on tangible interaction, novel fabrication techniques, and decentralized optical tracking.

Florian Echtler
Aalborg University, Aalborg, Denmark
floech@cs.aau.dk

Florian Echtler is associate professor in the Human-Centered Computing group at Aalborg University. His research interests are decentralized mobile interaction and ad-hoc creation of ubiquitous computing environments. Additional topics covered by his research include computer vision for HCI applications, sensor technology, and rapid prototyping.