GeoCAM: An IP-Based Geolocation Service Through Fine-Grained and Stable Webcam Landmarks

Qiang Li, Zhihao Wang, Dawei Tan, Jinke Song, Haining Wang, *Fellow, IEEE*, Limin Sun, and Jiqiang Liu, *Senior Member, IEEE*

***Abstract*— IP-based geolocation is essential for various location-aware Internet applications, such as online advertisement, content delivery, and online fraud prevention. Achieving accurate geolocation enormously relies on the number of high-quality (i.e., the fine-grained and stable over time) landmarks. However, the previous efforts of garnering landmarks have been impeded by the limited visible landmarks on the Internet and manual time cost. In this paper, we leverage the availability of numerous online webcams used to monitor physical surroundings as a rich source of promising high-quality landmarks for serving IP-based geolocation. In particular, we present a new framework called *GeoCAM*, which is designed to automatically generate qualified landmarks from online webcams, providing an IP-based geolocation service with high accuracy and wide coverage. GeoCAM periodically monitors websites hosting live webcams and uses the natural language processing technique to extract the IP addresses and latitude/longitude of webcams for generating landmarks at a large-scale. Given latency and topology constraints among webcam landmarks, GeoCAM uses the maximum likelihood estimation to approximately pinpoint the geolocation of a target host. We develop a prototype of GeoCAM and conduct real-world experiments for validating its efficacy. Our results show that GeoCam can detect 282,902 live webcams hosted in webpages with 94.2% precision and 90.4% recall, and then generate 16,863 stable and fine-grained landmarks, which are two orders of magnitude more than the landmarks used in prior works. To demonstrate the superiority of using large-scale webcams as landmarks, we implement four different geolocation algorithms and compare their performance between webcam landmarks and open-source landmarks. The evaluation results show that all the algorithms can significantly improve geolocation accuracy by using webcam landmarks.**

***Index Terms*— Geolocation, webcam, landmark.**

Manuscript received July 5, 2020; revised December 13, 2020 and March 26, 2021; accepted March 27, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C. F. Chiasserini. Date of publication April 28, 2021; date of current version August 18, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802804 and in part by the National Natural Science Foundation of China under Grant 61972024. The preliminary version of this article [45] was published in the Proceedings of The Web Conference 2020. *(Corresponding authors: Qiang Li; Jiqiang Liu.)*

Qiang Li, Dawei Tan, Jinke Song, and Jiqiang Liu are with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China (e-mail: qiangcas@gmail.com; jqliu@bjtu.edu.cn).

Zhihao Wang and Limin Sun are with the Institute of Information Engineering—Chinese Academy of Sciences (CAS), Beijing 100093, China, and also with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China.

Haining Wang is with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA.

Digital Object Identifier 10.1109/TNET.2021.3073926

I. INTRODUCTION

IP-BASED geolocation is used to determine the real-world geographic location of an Internet-connected host [1], which is valuable for many Internet applications, including targeted advertising, content delivery, and online fraud detection. IP-based geolocation involves mapping an IP address (or a domain name) to a country and region (city), as well as the corresponding pair of latitude/longitude. However, commercial geolocation databases only achieve less than 95.8% on the country-level and various disagreements on the city-level [2]. Nowadays, there is no official source of geolocation databases achieving high accuracy and wide coverage for Internet users.

Providing an accurate geolocation service is heavily dependent on the number of high-quality landmarks. Given sufficient landmarks, geolocation services would provide high accurate mappings between IP addresses and geographical locations. However, efforts to gather such mapping information are impeded by the limitations of available landmarks on the Internet. Those landmarks are community-based [3]–[5], leading to a limited scale in terms of their number and coverage. For instance, PlanetLab [5] only includes 420 available landmarks, and most of them are located in Europe and North America. They are mainly located on academic networks, with reduced availability on residential or commercial networks. Prior works proposed to increase the number of landmarks by mining web services [6], [7]. They assumed that web services are hosted locally. However, due to the ever-increasing popularity of cloud services and content delivery networks (CDNs), this assumption is no longer valid. In addition, dynamic IP addresses of those local web services would also lower their effectiveness.

As the pervasiveness of Internet-of-Things (IoT) systems, we have witnessed the significant increase of online webcams widely deployed for monitoring physical surroundings around the real world. Those online webcams are inherently associated with IP addresses and geographical locations, and more importantly, they are relatively stable over a long period, becoming ideal candidates for being used as landmarks. Thus, by leveraging the availability of numerous online webcams as promising landmarks, we are able to successfully address the challenging problem, i.e., the lack of high-quality landmarks, in IP-based geolocation.

In this paper, we propose a framework called *GeoCAM* that generates high-quality landmarks by automatically extracting the IP addresses and latitude/longitude of online webcams at a large scale. We first need to search for those websites that have gathered webcams and exposed their live streams to the public. Specifically, we utilize unique features in live streams

of webcams to find those websites and select the top 100 sites as the target websites for GeoCAM. After selecting target websites, GeoCAM periodically monitors these websites via crawling and uses the machine learning algorithm to determine whether a webpage includes a live webcam. Once a live webcam is found, we utilize the natural language processing technique to extract the latitude/longitude and geographical information of the device. Thus, the webcam landmarks generated by GeoCAM include the information of IP addresses and corresponding pairs of latitude/longitude. We implement three geolocation algorithms from previous research [8]–[10], and a hybrid algorithm of our own approach. Based on webcam landmarks, we can approximately pinpoint an individual host's geolocation with high accuracy and wide coverage.

To validate the efficacy of GeoCAM, we build a prototype and conduct real-world experiments. Our results show that GeoCAM can detect webcams from webpages with 94.2% precision and 90.4% recall. In total, we collect 282,902 webcams and generate 16,863 webcam-based landmarks, which are two orders of magnitude more than the landmarks used in prior works. These webcams cover around 170 countries, 6,450 cities, and 2,880 ASes. We compare webcam landmarks with open-source landmarks and commercial geolocation databases, respectively, demonstrating the superiority of using large-scale webcams as landmarks.

For the evaluation of IP-based geolocation services, we implement four different geolocation algorithms and compare their performance between webcam landmarks and open-source landmarks. Our evaluation results show that webcam landmarks can help all the geolocation algorithms achieve significant performance improvement. Note that webcam landmarks from third-party sources might contain errors. At present, our GeoCAM does not provide automatic calibration for error correction. We distribute the landmark dataset to the public at the website [11], and we will periodically update its calibrated version in the future.

The rest of this paper is organized as follows. Section II introduces the background of IP geolocation and webcam recognition. In Section III, we present the architecture of GeoCAM. Section IV describes the IP geolocation enabled by GeoCAM. In Section V, we perform experiments on 100 aggregator websites to find webcam landmarks and measure the sheer number, stability, and coverage of these landmarks. In Section VI, we describe GeoCAM geolocation performance, and compare webcam landmarks with commercial geolocation databases and open-source landmarks. Section VII discusses the limitations and privacy concerns of our approach. Section VIII surveys related works of IP geolocation, and finally, Section IX concludes the paper.

II. Background and Motivation

In this section, we present the background of live webcams available from online websites and the landmark generation based on those online webcams for IP-based geolocation.

A. Live Webcam

Webcams have been increasingly connected to the Internet in the past decade. A webcam is one typical Internet of Things (IoT) device [12], [13] for monitoring physical surroundings, such as a plaza, a street, an industrial area, or a home theater. For the sake of remote access and control, webcam devices are visible and accessible through their IP addresses. In contrast to web services, webcam devices are fixed in physical places and remain relatively stable over time. Thus, online webcams have great potential to be used as promising landmarks but have not yet been explored.

Fig. 1. An example of live webcam on websites.

Figure 1 shows one example, in which a live webcam hosted on the website of https://www.pictimo.com/pictimo.com, monitoring the surroundings at "Glenwood Springs hottub, United States". The website utilizes the web applet to host live streams of the webcam, which is labeled with HTML tag "<img src=IP address:port/mjpg", disclosing the webcam's IP address and corresponding port. Further, the webcam's latitude/longitude pair *(−107.340740, 39.527105)* is also embedded in the HMTL file. Therefore, we can extract a webcam's IP address and geographical information to generate a landmark for IP-based geolocation services.

B. Landmark Generation

A high-quality landmark implies that its IP address and corresponding latitude/longitude remain stable over time. Prior works [6], [7] utilize web mining to collect landmarks from web services for providing IP geolocation. However, their landmarks have become much less available to the public, due to the wide use of CDNs and clouds for hosting web services. In addition, the mappings between IP addresses and geographical information of such landmarks based on web services are not stable over the time.

For the generation of high-quality landmarks, two fundamental properties must be held for the candidates. First, landmark candidates should be relatively stable over the time; in other words, they should be publicly available for access without changing their IP addresses and geolocations for a long time period. Webcams are usually placed in fixed positions for a relatively long period except for being uninstalled. We will periodically monitor websites to update the webcam landmarks for guaranteeing their effectiveness (see details in Section III-B). Second, landmark candidates should include fine-grained geographical information for Internet geolocation services. Fortunately, we found many websites provide latitude/longitude and geographical information of those hosted webcams via user provision or manual inspection. Further, webcam images also disclose the surrounding physical information for fine-grained geographical information.

In this work, we propose GeoCAM to automatically extract geographical information and generate high-quality landmarks from live webcams hosted on websites. We leverage a set of existing resources and tools to address several practical problems in the process of GeoCAM, which are briefly introduced below.

Webcam Distribution Websites: Many websites collect webcam resources and distribute live streams to the public for multiple purposes, e.g., advertising beautiful scenery, monitoring

TABLE I
REGEX MATCHING FOR DETECTING AGGREGATION SITES

	Regex
JPEG	(?:CH=|Get(?:Data\.cgi|OneShot)|JpegCam \=|SnapshotJPEG|\-wvhttp\-01\-|axis \-cgi|c(?:amera|gi\-bin)|image(?:\.jpg)?|jpg|loginpas\ =|mjpg|pwd\=|resolution\=(?:640x480)?|snap(?: \.jpg|shot\.cgi)|vi(?:deo(?:\.(?:cgi|jpg|mjpg))?|ewer))
MJPEG	(?:CH\=|Get(?:Data\.cgi|OneShot)\-wvhttp\-01\- |a(?:ction\=stream|xis\-cgi) |c(?:amera|gi\-bin)|faststream\.jpg|image(?:\.jpg)?| mjpg(?:\.cgi)? |pwd \=|resolution\=|snapshot \.cgi|vi(?:deo(?:\.(?:cgi|jpg|mjpg)|stream\.cgi)))
VLC	(?:GetData\.cgi|axis\-cgi|c(?:amera|gi\- bin)|faststream\.jpg|mjpg|p(?:wd)? \=|resolution\=640x480|video(?:\.cgi)?)
FFMPEG	(?:cgi\-bin|pwd\=|resolution\=(?:640x480)?|user\ =admin|video\.mjpg)

traffic and weather. Considering various webpages organized by those websites, we need to filter out outlier webpages and keep useful webpages for geographical information extraction. We utilize the machine learning algorithms to build the classification model to determine whether a webpage is hosting a live webcam.

Information Extraction: For each webpage hosting a live webcam, we further extract the geographical information from the webpage, including IP address and latitude/longitude. In the Natural Language Processing (NLP) community, extracting such elements from a document is defined as Named Entity Recognition (NER) [14], which has been extensively studied. However, we cannot directly use NER to extract information because it is highly related to a specific domain. Geographical names usually are non-dictionary words, leading to low precision and recall. In this paper, we leverage a rule-based NER and local positions to extract landmarks from various webpages.

III. GEOCAM: DESIGN AND IMPLEMENTATION

In this section, we present the design and implementation of GeoCAM. Figure 2 illustrates its architecture, which consists of two major components: the webcam content picker (WCP) and the landmark generation (LG). The WCP first automatically scrapes webpages from websites, removing irrelevant content through the HTML parser. Cross-reference links are extracted to remove replicated and redundant pages among those webpages we collected. Further, the WCP utilizes machine learning algorithms to determine whether a webpage contains a live webcam. For each page considered to be webcam-relevant, the LG converts all its content (scripts, menus, and images) to texts, and preserves a snapshot of a live webcam. We use the regex to extract IP addresses, domain names, and geographical coordinates from webpages. The LG uses the rule-based NER to extract geographical names from various webpages and convert them to the latitude/longitude pairs. The LG outputs the landmark set, as a key-value format (IP, *(lat, lon)*). Note that we manually validate the effectiveness of a landmark by conducting a comparison with the stored snapshot of the corresponding live webcam in the Google Map. By creating a large number of landmarks, we can provide IP-based geolocation services. Below we elaborate on the details of GeoCAM.

TABLE II
FULL LIST OF 100 WEBCAM AGGREGATOR WEBSITES

Domain names of aggregator websites			
te.ua	cvlt.ch	krvn.com	pictimo.com
101.at	dczo.ch	ktru.org	seovisit.ru
1tv.me	ezpc.ru	mosgt.ru	turistik.cz
1vl.ru	fcdp.ro	phys.org	webkams.com
bay.tv	imao.sk	skjm.com	worldcam.eu
bye.ch	iski.it	syote.fi	earthcam.com
cic.ba	keri.ee	tveye.su	geocities.jp
nsf.se	lkvm.cz	tvjoy.ru	livecam.asia
och.nu	lszp.ch	tvway.ru	niki-surf.ru
ozi.ro	pike.jp	xa911.cn	racamera.com
r4n.it	rcnt.es	ipcams.ch	reklboard.ru
sld.uk	rftp.ru	mosday.ru	uamuseum.com
ufv.ca	seom.bg	telus.net	walltrend.ro
wff.lv	thai.mn	ttrix.com	webcambg.com
www.hr	trop.ch	dibcras.ro	camhacker.com
ycs.at	wdhd.ru	fboller.de	checkcams.com
9310.no	zvho.ch	lavrsen.dk	inchbeach.com
a2ch.ru	2ch.live	letunam.ru	opentopia.com
abgc.pl	56kb.com	sravni.org	roxy-world.ro
abvm.fr	camua.ru	webcams.bg	seo-surf.info
aset.no	etar.org	geoearth.ru	skidkamera.se
aupm.fr	fgsrb.ru	geometeo.it	skiweather.eu
bbox.ch	ilm24.ee	gobefore.me	top-kamery.cz
bowa.dk	ketry.cz	insecam.org	webkameror.se
bswr.de	kneb.com	panorama.sk	goowebcams.com

A. Websites

GeoCAM relies on the websites that are hosting a large number of live webcams [15]. We collected those websites using a heuristic rule: if a website distributes live streams under web applets (e.g., Joint Photographic Experts Group (JPEG) [16], Motion JPEG (MJPEG), VideoLAN media player(VLC) [17], and FFMPEG [18]), we keep it as a candidate site. Table I lists the regular expressions of keywords related to web applets for finding websites that host live webcams. Note that embedded libraries are the most popular web applet to load live streams of webcams. We leverage a Common Crawl to gather websites with live webcams, an open repository of web crawl data provided by a non-profit organization [19]. The Common Crawl fetches webpages on the Internet every month and its datasets are available to the public through Amazon S3. To the best of our knowledge, it is the largest open-source repository for webpages, consisting of billions of webpages with raw data and metadata, where archive files are stored in the Web ARChive (WARC) format. We used the keywords through the Common Crawl and the Google search engine, and discovered more than 3,000 websites. Then, we manually selected 100 websites (listed in Table II) that host a large number of webcams and used them as the GeoCAM's input. One might concern that the limited number of websites could hinder the scalability of our approach. However, based on those 100 websites, we have already found 16,863 visible and accessible webcam landmarks, which are two orders of magnitude more than those used in the existing services. If a new site is available in the future, our approach can generate new landmarks with little modifications.

B. Relevant Webpage Detection

Aforementioned, to automatically generate landmarks from websites, we first need to scrape webpages, filter out noise and irrelevant content, and determine whether a live webcam is running on a webpage.

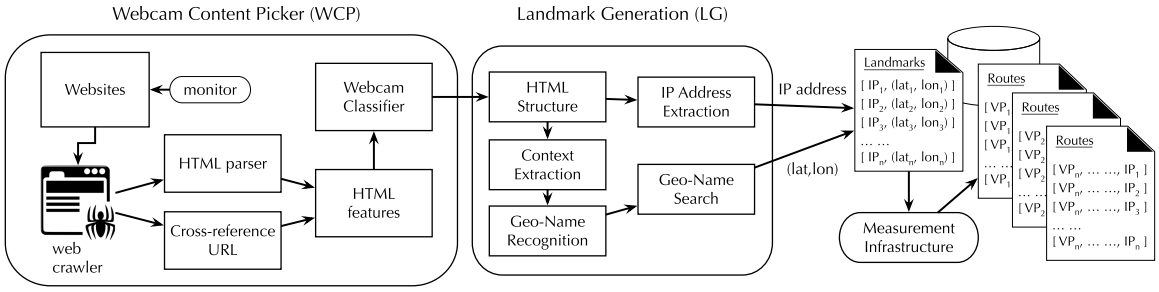


Fig. 2. The architecture of *GeoCAM*.

Web Crawler: Our web crawler is designed to periodically monitor a website to collect its webpages. Since different websites have different templates and HTML structures, we design a web crawler to scrape webpages from those sites. Given a website, the WCP first utilizes the web crawler to obtain all its pages through the breadth-first search. Specifically, we parse the website homepage to explore all its URL links, and iteratively parse the URL links to explore the next-layer pages, until no more links are found.

A problem here is that websites might change webpages or live webcams over time, e.g., new pages are added or old pages are removed. However, we believe that webcams are relatively stable for a long time period. The WCP periodically monitors websites/webpages by re-accessing those URL links to identify whether they are still available. In practice, the monitoring period for websites is one month and the period for webpages is one week.

Pre-Processing: Webpages collected by the web crawler involve irrelevant information, such as advertisements, icons, and navigation bars. The WCP utilizes the HTML parser to remove those irrelevant elements from webpage files, e.g., <ad> and <icon>. Websites usually utilize Javascript and Frame in HTML to post live streams of webcams. We keep the content of webpages together with javascript and frame for extracting webcam information. In addition, there are many cross-reference links in those webpages. Some URL links belong to a same site, while some come from different websites. If a link involves a page we have collected, we remove the duplicated one. If a link comes from different sites, we extract its domain name as a candidate website. We manually inspect those candidate sites to determine whether they host a large number of webcams. After scraping and preprocessing, we store all HTML files and use URL links as their index.

Webcam Classification: Another problem here is that some webpages do not post any live streams, while some include multiple webcams. Hence, we divide webpages into three groups: *none*, *single*, and *multiple*. In the first group ("none"), many webpages belong to user-generated-content (UGC) pages or others (e.g., contact pages, introduction pages, or login pages). We cannot extract any landmark information from those webpages that have no webcams. In the second group ("single"), webpages have only one embedded live webcam and distribute its live streams to the public. We can generate a landmark from such a webpage, including its IP address and latitude/longitude. In the third group ("multiple"), webpages might have more than one webcams. Generally, there are two kinds of HTML templates for multi-webcam pages: (1) a page displays multiple webcams under a particular topic, and (2) a page shows a single webcam together with some recommended webcams from the website owner. We can generate multiple landmarks from those webpages.

TABLE III
REGEXES FOR EXTRACTING FEATURES

General type	Regex
IPv4	((25[0-5]—2[0-4][0-9]—[01]?[0-9][0-9]?)\.){3}(25[0-5]—2[0-4][0-9]—[01]?[0-9][0-9]?)(/([0-2][0-9]—3[0-2]—[0-9]))?
Domain Name	https?://[-a-zA-Z0-9._:%\+~#=]{2,256}\.([a-z]{2,6}—\d{1,3})\b([-a-zA-Z0-9._:%\+~#=?&;\(\)\[\]]*)
Geographical Coordinate	(latitude(\n—.)*?(?P<lat>[-+]?\d{1,2}\.\d+)(\n—.)*?longitude(\n—.)*?(?P<lon>[-+]?\d1,3\.\d+))—(longitude(\n—.)*?(?P<lon>[-+]?\d1,3\.\d+)(\n—.)*?latitude(\n—.)*?(?P<lat>[-+]?\d{1,2}\.\d+))

To automatically partition webpages, we leverage the observation: a live webcam is encapsulated as the specific HTML element to post its snapshot or live stream on webpages. The WCP extracts webcam features to infer the classification model of webpages as follows:

1) IP address and domain name. To post a live stream, a website needs a webcam's source path to load its video content. Some sites directly use the webcam's IP address and port number, while others use a domain name to host the webcam's content.
2) Snapshot. Webpages provide a webcam snapshot as its content to the public. The update rates of snapshots are different according to different website configurations. Typically, Joint Photographic Experts Group (JPEG) and Motion JPEG (MJPEG) are used to update the webcam's snapshots.
3) Live stream. Webpages directly distribute live streams of webcams to the public. VideoIAN media player (VLC) and FFMPEG are usually software libraries to load live streams.

To present those features, we use the regex matching to extract those features, as listed in Table III. A binary vector is to indicate the presence of those features in a webpage. If the regex finds a character, the corresponding factor in the vector is set to 1, otherwise it is to 0. We use the machine learning algorithms to derive the classification model of the webpage. The model outputs are with three class labels: *none*, *single*, and *multiple* (see details in Table VI).

Implementation: We use BeautifulSoup [20] to parse each HTML file into a list of chunks based on the HTML tags. For each chunk, we use regexes to extract IP addresses, domain names, snapshots, and live streams. We use scikit-learn [21]

Algorithm 1 Multiple Webcam Extraction

Input: $HTML$, webpage with multiple webcams
Output: $Results$, IP/Geo information of webcams
$Candidates \leftarrow$ {HTML tags within webcam features}
$Results \leftarrow []$
$dict \leftarrow \{\ \}$ ▷ store (parent tag, children Set)
while $length(Candidates) > 0$ **do**
  for all $x \in Candidates$ **do**
    $parent \leftarrow$ parent tag of x
    Append(x, $dict[parent]$)
  end for
  if all candidates share same parent element **then**
    for all $x \in Candidates$ **do**
      $< ip, geo > \leftarrow$ IP/Geo from x
      Append(<ip, geo>, $Results$)
    end forreturn $Results$
  else
    for all (i, Set) pairs $\in dict$ **do**
      if $len(Set) > 1$ **then**
        remove $\forall$ tags $\in Set$ from $Candidates$
      else
        replace Set with the tag i
      end if
    end for
  end if
end while

to implement Support Vector Machines (SVM) algorithms. We choose the radial basis function (RBF) as SVM kernel to learn the classification model. We deploy the classification model in GeoCAM. For single- and multi- webpages, GeoCAM further extracts geographical information for landmarks.

C. Landmark Information Extraction

Once a webpage is tagged as containing a webcam, we need to extract relevant context information from the webpage, especially the information essential to a landmark: its IP address and latitude/longitude pair. Below we present the details of the LG component for generating landmarks.

1) Webcam Identification: As we mentioned earlier, we can generate multiple landmarks from those webpages that host multiple webcams. For multiple webcam pages, we divide those webcams into different entities for extracting geographical information.

We use Algorithm 1 to extract different webcam entities from multiple webcam pages. In the beginning, we extract all HTML tags involving relevant context information (e.g., IP address/port, domain names, and latitude/longitude) as the candidate-tag set. For each HTML tag with a webcam feature, we record the tag and its parent tag. We use the dictionary set $dict$ to store those tags in the format of {parent tag, children set}. If two tags have the same parent tag, we append them into the same children set. Then, we determine the type of multiple webcam pages according to the structures of those candidate tags. Specifically, we partition webcam entities based on the following rule. If all tag items have the same parent tag, the multi-webcam page is probably used to show several webcam entities under a particular topic. We extract the geographical information from all webcam entities, including the IP address and geolocation information. If those tags cannot be merged into the same root, the multi-webcam page is probably used to host a single webcam together with other recommended webcams from the website owner. We do not take recommended webcams into consideration because they are duplicated with webcams in other webpages. Note that recommended webcam tags share the same parent tag in the HTML list. We repeatedly remove these tags from the candidate set until reaching the root tag in the recommended webcam list.

2) The Regex Extraction: We use the regex to extract the webcam related information from webpages, including IP address and port, domain name, and geographical coordinate. They have distinctive character features, and the regex can achieve high accuracy in practice, as shown in Table III.

IP Address and Port: When an IP address and port number are directly exposed on the webpages, we use the regex to extract them from the HTML file. Once an IP address is found, the LG generates a candidate landmark that is formatted as *[IP, (null, null)]*. Note that not all webcams expose their IP addresses and ports on webpages.

Domain Name: Some webcams use web services to host live streams of webcams and hide real IP addresses. Web services registered their domain names and post their URL links for webcams. The LG extracts URL links or corresponding redirection links, and sends them to the WCP for scraping HTML files. If an IP address is extracted from a new HTML file, we need to validate its effectiveness. The rule is straightforward: if their heads (<head> and </head>), or titles (<title> and </title>) are the same, the IP address is identified as the webcam address.

Geographical Coordinate: The geographical coordinate refers to a *(longitude, latitude)* pair, which is used to directly present the landmark geographical context. Note that not all webpages include geographical coordinates for webcams. There are two places to store geographical coordinates, including the HTML body and Frame. Frame is a common place to embed a webcam's geographical coordinate, e.g., Google Map. We directly use the regex (Table III) to extract a webcam's latitude/longitude.

3) Geographical Name Recognition: Many webpages only expose geographical names for webcams and their latitudes/longitudes are not available. For example, the live stream on site https://www.pictimo.com/pictimo.com (Figure 1) shows the geographical name "Glenwood Springs hottub, United States". We need to extract those geographical names, and convert them into the latitude/longitude pairs. There are two problems for directly using NER for identifying geographical names, leading to low precision and recall. First, webcam entity may have several geographical contexts, but they expose at different coarse levels, including non-normalized and general forms. Second, many geographical names use non-dictionary and non-English words, creating various name entities.

We propose the rule-based geographical name recognition (rGNR) to extract those geographical names from webpages. Figure 3 illustrates the overview of rGNR in the process of GeoCAM. The rGNR leverages the observation that many name entities are stored in fixed positions, including <title>/<meta>, URL, and images. The <title> and <meta> of a webpage usually contain a location description of a webcam. URL is an interesting place, where a webpage indicates

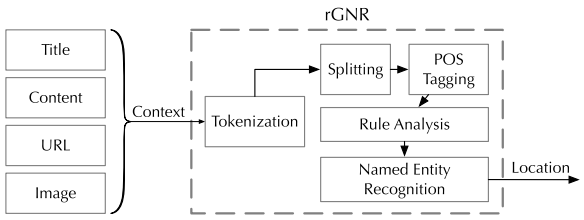


Fig. 3. The rule-based geographical name recognition (rGNR).

the location information as a part of the URL. The reason is that a website might organize its webpages in a hierarchical structure based on the geographical name. Images or snapshots of webcams have been embedded with relevant location and timestamp information, e.g., the example in Figure 1. We only extract the content from those places as the rGNR input.

We convert sentences from those places into tokens. They are connected through *delimiters*, include "\" and webcam characters. We utilize the POS tagger to split sentences into different segments. POS taggers have four types of context, including capitalized characters, webcam characters, location words, and delimiters. Domain names and titles might be capitalized characters. Webcam characters usually appear together with geographical names in the same sentence, e.g., "webcam", "kamera"(Cezch, Indonesian), and webcam vendors. Location words are the textual descriptions before or after a specific geographical name, e.g., park or harbor. After segmenting, we apply the name entity recognition (NER) [22] to recognize those geographical names.

Once a geographical name is recognized, we convert the geographical name into the corresponding latitude/longitude pair. Here, we use the public geoname database to find its geographical coordinate. For providing IP-based geolocation services, the LG generates the landmarks that are formatted as the [IP, (lat, lon)].

Implementation: For webcam images, we use an optical character recognition engine Tesseract [23] which recognizes the text from an image. However, its performance is low due to low quality of images and non-dictionary words under different languages. For textual descriptions on webpages, we use the POS tagger tool [22] to obtain segments from webpages. We further use rGNR to recognize geographical names on webpages. We use open source geoname database provided by OpenStreetMap [24], a free online map service with over 20 million names and corresponding coordinates, to build geoname-coordinate mappings.

IV. IP Geolocation Computation

The objective of a geolocation service is to determine an IP host's physical location. The related IP geolocation computation involves two major aspects: landmark-based probing and geolocation algorithms. The landmark-based probing is to obtain the latency and network topology constraints of the target and landmarks. The geolocation algorithm is to estimate the distance between the target and landmarks based on measured latencies and then calculate the coordinates of the target. In this work, we use three geolocation algorithms, including Constraint-Based Geolocation (CBG) [8], Octant [9], Spotter [10], and our GeoCAM. Since we cannot find the open-source code for those three algorithms, we have to re-implement them based on the descriptions presented in previous work.

A. Landmark-Based Probing

As we mentioned before, GeoCAM can generate a large number of webcam landmarks, which obviously improve geolocation accuracy. However, the measurement cost is high because our server must send packets to all landmarks and wait for replies. Further, if we do this simultaneously, the extra network traffic may induce congestion and then cause the inaccuracy of the latency measurement. To address this challenge, we use a two-stage measurement process proposed by prior works [25], [26].

First, we identify the coarse region of the target host. We simply estimate the coarse-grained region based on the delay-distance relationship. We use the following equation to compute the distance between the probing server and the target host:

$$d(i,j) = T(i,j) * c * f,$$

where $T(i,j)$ is the latency between two nodes i and j, c is the speed of light in vacuum (3×10^8), and factor f is a self-calibration coefficient, which is set to $2/3$. Once we know the coarse region of the target host, we select the landmarks on this region and those landmarks that are far away from the target are less useful. Then, based on the selected landmarks on the course region, we select a larger group of candidate landmarks to approximately pinpoint the target host by using geolocation algorithms.

Measurement Tools: Given webcam landmarks and target hosts, we use the measurement tools to obtain their network topology constrains and the time delay. In our experiment, the traceroute TCP on a commonly used port has the highest success rate. Thus, we use the traceroute [27] to measure the latency and network topology, as well the routing path between our probe server and targets. In addition, we can obtain the routing path through the traceroute and figure out the common routers between the landmarks and the target host.

Vantage Point (VP): A VP is one of our servers with the known geographic location, which sends packets to the target and wait for the responses. The number and distribution of vantage points (VPs) are vital for inferring the target's region. Once VPs are fixed, we use them to send the traceroute probing traffic to measure the latency and network topology constraints between the target and landmarks.

B. Geolocation Algorithms

Besides the geolocation algorithms of GeoCAM, we also implement three geolocation algorithms from prior works for comparison purposes.

1) Constraint-Based Geolocation: Gueye *et al.* [8] proposed the CBG that builds a linear programming model between the distance and the latency, i.e., $d_i \sim a * t_i + b$. For a target host, CBG measures the latency and uses the linear model to calculate its distance. Noted that the distance is limited by the speed of the signal propagation in fiber-optic cable. CBG uses the distance to obtain a coarse-grained region for a target IP address. Further, CBG can utilize landmarks either from our GeoCAM or other accessible sources to multilaterally locate a fine-grained region for the target host. In our implementation, the webcam's latitude/longitude is the circle center, and the estimated distance by CBG is the radius. The intersection of these circle regions is the geographic range of the target host.

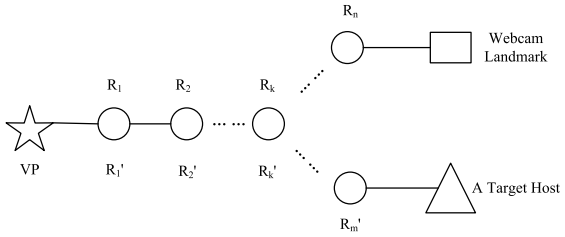


Fig. 4. The network topology constraints for GeoCAM.

2) Octant: Wong *et al.* [9] proposed the Octant that provides upper bound and lower bound curves for representing the range of latency-distance relationship. Specifically, the Octant uses the convex hull to learn the smallest convex set that contains the points (latency, distance) in two dimensions. For every targeted host, the Octant utilizes the curves to calculate the maximum and the minimum distances, up to 50% and 75% of all round-trip times, respectively. Similar to CBG, the Octant uses landmarks to multilaterally locate a coarse-grained region for a target IP address.

3) Spotter: Laki *et al.* [10] proposed the Spotter that uses the polynomial model to fit the function of the delay-distance relationship. Specifically, it uses the quadratic or cubic polynomial to learn the regression model. Given a latency from a landmark or target, the Spotter calculates its distance as a Gaussian distribution with mean μ and standard deviation σ. Similar to CBG and Octant, the Spotter also uses landmarks to multilaterally geolocate a predicted region for a target IP address.

4) GeoCAM: The geolocation algorithm of GeoCAM is a hybrid that integrates the Spotter and network topology constraints. Given a target host, we geolocate it in the following three stages: (1) using traceroute, VPs measure the RTTs to landmarks and the target; (2) we identify the shared routers and calculate the distance between landmarks and the target; and (3) we utilize the maximum likelihood estimation to approximately locate the target. Below we elaborate on the details of the last two stages.

The inflated latencies and indirect routers could introduce errors for the function of the delay-distance relationship. We utilize the network topology constraints to reduce errors caused by active measurements. Figure 4 depicts the network topology for estimating the latency and distance. The vantage point (VP) is the probing server with the known geographic location, which sends packets to the IP hosts. The routing path between the VP and the webcam landmark is denoted as (R_1, R_2, …, R_n), and the path between the VP and the target host is (R'_1, R'_2, …, R'_m). We search for the same set of routers shared by the two paths and locate the same router that is the closest to the two destinations, denoted as $R_k = R'_k$. The sum of $T(R_k, L)$ and $T(R_k, H)$ presents the latency between the webcam landmark and the target, where T is the latency, R_k is the same router closest to both destinations, L is the landmark, and H is the target.

We use the maximum likelihood estimation to derive the latitude/longitude of the target host. First, we calculate the conditional probability for the geographical distance and latency. We use the set of VPs, denoted as $\boldsymbol{V}$, and the set of webcam landmarks ($\boldsymbol{L}$) to locate the target host. We calculate the relative latency of any two landmarks in the target region as follows:

$$rT_{ij} = T(V, L_i) + T(V, L_j) - 2T(V, R_k), \tag{1}$$

where $L_i, L_j \in \boldsymbol{L}$. Then we use all the relative latencies of landmarks in the target region to infer the conditional probability of distances. Here we use the truncated normal distribution to represent the conditional probability as follows:

$$p(d|rT) = \frac{1}{\Phi(\mu/\sigma)} \cdot \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot \exp\left(-\frac{(d-\mu)^2}{2\sigma^2}\right),$$
$$\sigma = \sigma(d|rT), \mu = \mu(d|rT), \tag{2}$$

where rT is the latency between any pair of landmarks and $\Phi(\mu/\sigma)$ is the cumulative distribution function of normal distribution. For each conditional probability, we utilize the log-likelihood function to compute the maximum likelihood function as follows:

$$L_i = \sum_{i=1}^{K} \log\left(P(d(x, L_i)|rT_{mi})\right). \tag{3}$$

To estimate the target location x_m that maximizes L_i

$$\hat{x}_m = \arg\max_{x \in C} L_i(x), \tag{4}$$

we choose the maximum likelihood probability to compute the latitude/longitude of the target host.

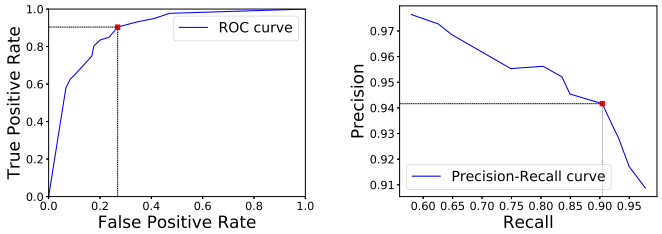
V. Landmark Experiments

In this section, we validate the efficacy of webcam landmarks created by GeoCAM based on two datasets.

A. Settings

We implemented a prototype of GeoCAM and ran it to automatically analyze 1.9 million webpages, using Ubuntu 14.04 on an AWS server with two 3.1GHz Intel Xeon Platinum 8175 vCPU, 8GB memories, and 10Gbps bandwidth. Here we detail the datasets used in this study.

Datasets: (1) We used *a labeled dataset* for training/testing the classification model and our geographical name recognition. This dataset contains 2,300 webpages and their corresponding webcams. These live webcams were manually collected from websites such as *pictimo*, *goowebcams*, *racamera*, and *insecam*. For every page, we manually provided its label, including *none*, *single*, and *multiple*. About 1,600 of them are tagged as *single*, 300 of them are tagged as *multiple*, and the rest (400) are tagged as *none* web pages. Each of them was manually checked to ensure that they were accurately extracted by GeoCAM. (2) We used a large-scale dataset from 1.9 million webpages to further validate the effectiveness of webcams for being used as landmarks. We manually extracted these popularity websites that are hosting live webcams on the Internet (see Table II for details). On those websites, we ran the GeoCAM to crawl 6 times in total (see details in Figure 6), on May 11, 2019, October 5, 2019, October 9, 2019, October 10, 2019, October 11, 2019, and October 12, 2019, respectively.



(a) ROC curve. (b) Precision-Recall curve.

Fig. 5. ROC and PR curve of webpage classification.

TABLE IV
COMPARE GEOLOCATION ENTITY RECOGNITION RESULTS OF GEOCAM AND CORENLP

Website	Language	False Negative Rate	
		GeoCAM	CoreNLP
goowebcams	English	4%	53%
racamera	Russian, English Polish, etc.	0%	76%
insecam	English, Chinese Russian, etc.	0%	48%
pictimo	English	0%	84%

B. Performance

Webpage Classification: We first evaluate the classification model's performance using the labeled webcam dataset. For the multi-class classification, the SVM algorithm with RBF kernel derives the boundary between one class to other classes. We divide the dataset into the training set (1,610 webpages) and the test set (690 webpages). We use false positive rate (FPR), precision, and recall to measure its performance, where FPR is the ratio of $|FP|/|FP+TN|$, the precision equals to $|TP|/|FP+TP|$, and the recall is $|TP|/|TP+FN|$. TP is the number of true positives, FN is the number of false negatives, FP is the number of false positives, and TN is the number of true negatives. Figure 5a shows the ROC curve of the TPR and FPR, and Figure 5b illustrates the precision-recall (PR) curve of the webcam classification performance. Our prototype achieves a precision of 94.2%, a recall of 90.4%, and a FPR of 21.4% in determining whether a webpage has a webcam. In practice, GeoCAM performance is acceptable for classifying web pages with webcams.

Webcam Location Extraction: We then evaluate the webcam location extraction of GeoCAM. We use two methods: (1) our proposed GeoCAM and (2) a general NER method called CoreNLP [22]. For each webpage, we manually extract the geographical location as the ground truth. Table IV lists the performance of GeoCAM and CoreNLP. The CoreNLP has a high false negative rate (FNR) on geographical name extraction. For instance, its precision is only 24% on the site racamera.com. By contrast, GeoCAM achieves very promising performance for extracting geographical names.

Overhead of GeoCAM: We first conduct experiments to measure the time cost of GeoCAM for generating webcam landmarks. Our GeoCAM prototype runs on a desktop computer (MacOS 10.14.4, 4vCPU, 8GB of memory, 64-bit OS). The GeoCAM process runs in a single thread. Table V lists the average and standard deviation of the GeoCAM's time cost. The WCP component takes 5.8 ms to scrape and process a webpage, and the LG component takes 22.68 ms to process a live webcam. The geolocation computation of GeoCAM costs 102.12 ms to calculate the geolocation information of a target host. We further compared the time cost between the traceroute-based approach and the ping-based approach. In our controlled experiments, we selected 200 webcam landmarks, and the average overhead is 1,218.77ms for the traceroute method. For the ping-based approach, the average cost is 271.63ms. Overall, the time cost of GeoCAM is low in practice, and we could further reduce the time cost by running GeoCAM in multiple threads.

TABLE V
AVERAGE TIME COST AT DIFFERENT STAGES FOR GEOCAM

Stage	Average time (ms)	Std Deviation
WCP	5.84	5.86
LG	22.68	101.23
Geolocation Computation	102.12	66.07

TABLE VI
THE NUMBER OF WEBPAGES COLLECTED BY GEOCAM OVER 100 WEBSITES

Stage	Webpage Number
Before filtering out	1,913,277
After filtering out	282,920
None-webcam hosting	1,630,357
Single webcam hosting	256,210
Multi-webcam hosting	26,692

The traffic overhead on webcams is limited to the geolocation stage. In the discovery stage, we use the web crawler to scrape the landmark information from aggregation websites. Thus, there is no traffic overhead imposed on webcams during the discovery stage. In the geolocation stage, we use webcams as landmarks. We introduce The extra traffic load on the webcam because of performing traceroute. The induced traffic overhead on the webcam is negligible, with only one traceroute request per VP, which does not cause any negative impact on the normal operations of webcams.

However, the traffic cost would not be neglected if every single VP used all the landmarks for locating a target. The selection of suitable landmarks and VPs before conducting traceroutes (preprocessing) is an effective way to reduce webcam landmarks' traffic overhead. We briefly describe the heuristic landmark selection as follows: (1) beforehand as preparation, we collect the ASes at the city-level and the traceroutes to all landmarks; (2) for a given targeted IP, we first traceroute the IP to determine its AS number and approximate city-level, and then we select a group of suitable landmarks for further geolocating the IP address. As a simple example, for a given IP address (217.128.7.5) in France Neufch, we are able to narrow down 16K landmarks into less than 100 for its geolocation search.

C. Webcam Landmark Validation

Landscape: Table VI lists the number of webpages collected by GeoCAM over 100 websites. In total, we collected 1,913,277 webpages. After filtering out unnecessary pages by WCP, there are 282,902 webpages remaining. Among those webpages, there are 256,210 pages hosting single live webcam, and 26,692 pages hosting multiple webcams. We observe that nearly 1.5 million webpages are not associated with live webcams, and thus we drop them out from the candidate set.

TABLE VII
RELEVANT INFORMATION EXTRACTION FROM WEBCAM-RELATED WEBPAGES

Item	Number
Live streams of webcams	216,974
[Latitude, Longitude]	57,909
Geographical names	187,119
IP addresses of webcams	378,899

The Number of Webcam Landmarks: Table VII lists the amount of relevant webcam information extracting from webpages. In total, the GS component extracts 378,899 IP addresses, 187,119 geographical names, 57,909 latitude/longitude pairs. It finally has 216,974 webcam candidates with IP addresses and geographical information (either latitude/longitude or geographical name). A landmark involves an IP address and its latitude/longitude, which we use a key-value pair to store. Note that those webcams might be overlapped with one another. The reason is that websites might scrape webpages from others. For instance, web-online24.ru and tvway.ru both collect webcams from hotel-novoros.ru. We use their IP addresses to filter out the duplicated ones. In other words, if two webcams have the same IP address, we remove the duplicated one. We obtain 16,863 landmarks with unique IP addresses and accurate latitude/longitude information.

Webcam Landmark Stability: To validate the stability of webcam landmarks collected by GeoCAM, we measure the number of available webcam landmarks and webpage URLs along with time. We illustrate their dynamic changes in Figure 6 using the datasets collected on May 11, 2019 and October 5, 2019, as well as the most recent datasets collected from October 9, 2019 to October 12, 2019 for four consecutive days. We can see that the number of available webcam landmarks remains stable even after 5 months (80.45%), with much less variations within one week, indicating their long-term stability. We observe that only 8.55% of webcam landmarks change their IP addresses every 24 hours.

Moreover, we describe a landmark as the tuple {webpage URL, webcam IP, GEO info}. We find that when a webpage URL is stable, the IP and geoinformation never change. We further discover the main cause of a landmark's IP change, i.e., webcams might get offline occasionally. At two consecutive observations, around 8% of webcams get offline and might get back next time. Hence, the large drop of available landmarks between May 11, 2019 and October 5, 2019 is caused by some webcams becoming offline. Another reason is the dynamic nature of IP addresses, and a webcam might not always be tied to a specific IP address.

Landmark Coverage: We conduct the analysis on webcam landmark coverage, including geographical distribution, AS distribution, and domain name distribution.

(i) Geographical Coverage. We first compare the geographical coverage of our landmarks with others. Figure 7 depicts the world map, where the blue dots represent our webcam landmarks, and the red dots are the landmarks from other platforms (Planetlab [5], PingER [4], and PerfSONAR [3]). As an example, Figure 8 depicts a more detailed geographical distribution of landmarks in the UK and Ireland. We can see that our webcam landmarks almost cover all geographical places of existing landmarks from others with much higher density. Europe and North America are the traditional regions that open source landmarks cover well, but the other regions are rarely covered. By contrast, our webcam landmarks well

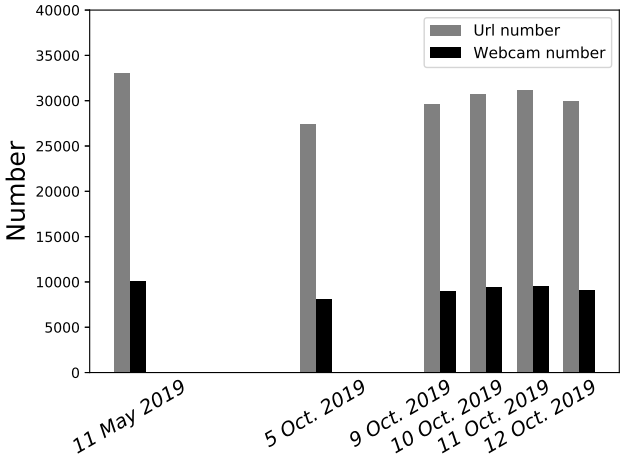


Fig. 6. Dynamic changes of URL/webcam landmark along with time.

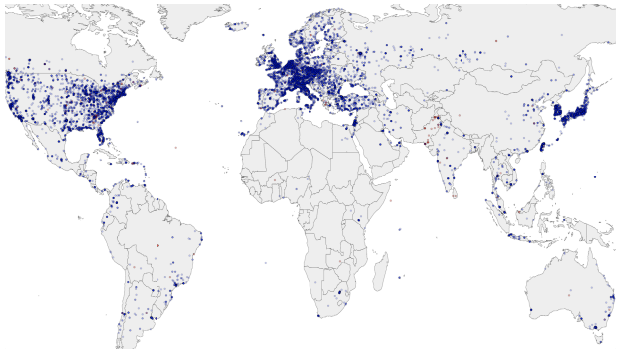


Fig. 7. Geographical distribution of webcam landmarks over the globe.

TABLE VIII
THE TOP 10 COUNTRIES OF WEBCAM LANDMARKS

Country	Number	Country	Number
USA	4,277	Turkey	584
France	1,001	UK	529
Italy	947	South Korea	424
Japan	864	Czech	392
Germany	631	Netherland	387

cover much more geographical areas, including Russia, South America, Turkey, India, and China. Interestingly, Japan has the highest density of webcam landmarks.

Here we briefly describe the country/city distribution for geographical coverage of webcam landmarks. In total, webcam landmarks cover 170 countries and 6,448 cities. About 25% of webcams are from North America and Europe. Table VIII lists the top 10 countries that webcam landmarks cover, and those countries are from North America, Europe, and Asia. We also list the top 10 cities covered by webcam landmarks, as shown in Table IX.

(ii) AS Coverage. We also analyze the autonomous system (AS) coverage of webcam landmarks. We build *block-asn* mappings based on the existing BGP routing table analysis data [28] supported by APNIC. Table X lists the top 5 ASes covered by webcam landmarks, where most of webcams are from residential networks. In total, we find that webcam landmarks cover nearly 2,875 ASes, much more than open source landmarks that are centered on academical networks.

(iii) Domain Name Coverage. Domain names of hosts are usually helpful to infer geolocation information. For instance,

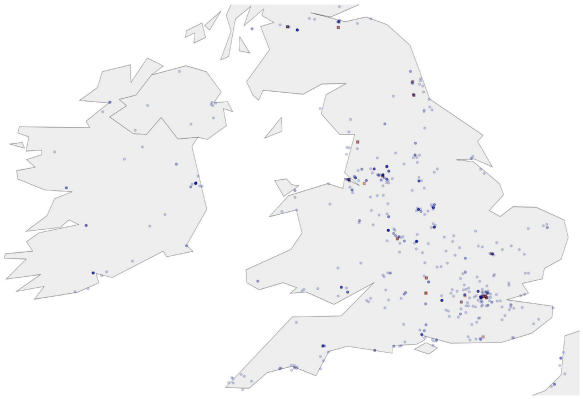


Fig. 8. Geographical distribution of landmarks in the UK and Ireland.

TABLE IX
THE TOP 10 CITIES OF WEBCAM LANDMARKS

City	Number	City	Number
Woods County	172	New York City	55
Sant'Agnello	110	Forlì del Sannio	53
Shintoku	105	Seoul	47
London	65	Shinjuku	44
Kostanay	59	Bangkok	41

TABLE X
TOP 5 ASES COVERED BY WEBCAM LANDMARKS

AS ID	AS network	Number
AS7922	COMCAST, US	774
AS9121	TTNET, TR	507
AS4766	KIXS-AS-KR Korea Telecom, KR	393
AS3269	ASN-IBSNAZ, IT	390
AS22394	CELLCO, US	364

from a hostname of *admin.umass.edu*, one can infer that the corresponding node belongs to the University of Massachusetts at Amherst. Therefore, we further analyze the domain name information of webcam landmarks. We use reversed DNS lookups to obtain domain names associated with IP addresses from our webcam landmarks. We resolve webcams' IP addresses to collect their pointer records (PTR). Note that not all IP addresses have a reverse entry for PTR. For 16,863 landmarks, there are 11,950 IP addresses with PTR records but 4,913 of them without PTR records. We observe that 311 PTR items have geographical clues, where 309 have the `edu` TLD and 9 have the `gov` TLD. When a record ends with a distinguishable TLD (`edu` or `gov`), we can infer their location information.

VI. GEOLOCATION PERFORMANCE

In this section, we use the webcam landmarks to provide geolocation services for approximately pinpointing the geolocation of an Internet host. More specifically, we first compare webcam landmarks with open-source landmarks and commercial geolocation databases, respectively, in terms of coverage and accuracy. Then we further apply webcam landmarks with four different geolocation algorithms and compare its performance with that of open-source landmarks.

Dataset: We randomly selected 120 targeted hosts in Europe from our webcam dataset. We manually inspected those snapshots of live streams of webcams, and searched relevant geographical contexts in Google Map to obtain their ground truth latitude/longitude information. Thus, every host has an IP address and corresponding latitude/longitude information. We use them as the targeted hosts with ground truth labels. For every IP address, we use 10 VPs across the U.S. and Europe.

TABLE XI
THE NUMBER OF VISIBLE AND ACCESSIBLE LANDMARKS

Data source	Num.	Scope
GeoCAM	16,863	Academic & Residential
PlanetLab [5]	420	Academic
PerfSONAR [3]	642	Academic
PingER [4]	127	Academic
RIPE Atlas [29]	458	Residential

A. Landmark Comparison

Open Source Landmarks: We compare our webcam landmarks with open source ones, including Planetlab [5], PingER [4], PerfSONAR [3], and RIPE Atlas [29]. Table XI lists the number of visible and accessible landmarks. Note that we only incorporate those visible and accessible landmarks. For instance, PerfSONAR [3] states that it has 2,143 nodes; however, most of them has become unavailable, due to usage restrictions on these nodes, and only 642 of them are still accessible. Moreover, those landmarks are from academical communities, rarely from other communities. Our webcam landmarks distribute in various types of places with a high geographical resolution, including residential and other environments. Overall, our webcam landmarks are two orders of magnitude more than the landmarks used in prior works.

As we mentioned before, when some landmarks are far away from the target, they become useless for geolocating that target. We measure the physical geographic distance between landmarks and all the target hosts that we selected. We compute the physical distance between two points (latitude, longitude) based on the geopy [30], which utilizes a spherical model of the earth with mean earth radius (approx 6,371 km) defined by the International Union of Geodesy and Geophysics. Note that effective measurements are more likely from landmarks close to the target. As shown in Figure 9, the gray-color bar is the number of webcam landmarks, and the black-color bar is the number of open-source landmarks. Obviously, there are much more webcam landmarks than open-source landmarks. In geolocation algorithms, the landmarks close the target are used to produce the predicted/estimated region for the target host.

Commercial Geolocation Database: We further compare our webcam landmarks with popular commercial geolocation databases, including IP2Location [31] and Maxmind GeoLite2. Figure 10 shows the CDF curve of deviation distances between our landmarks and commercial geolocation databasess. There are 44% webcam landmarks having the deviation less than 10km, compared with IP2Location. Note that the commercial databases cannot provide ground truth labels for our landmarks because those databases only provide coarse mappings between physical locations and IP hosts.

B. Geolocation Comparison

To demonstrate the superiority of using large-scale webcams as landmarks, we apply webcam landmarks with four different geolocation algorithms and compare the geolocation accuracy with that of open-source landmarks. Thu, we conduct four geolocation comparison experiments: (1) GeoCAM

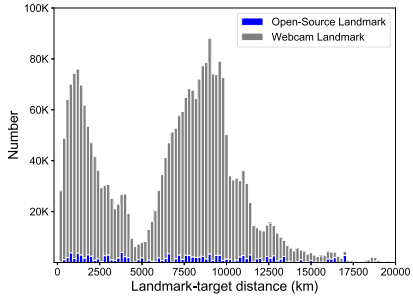


Fig. 9. The landmark number along with the distance between landmark and target.

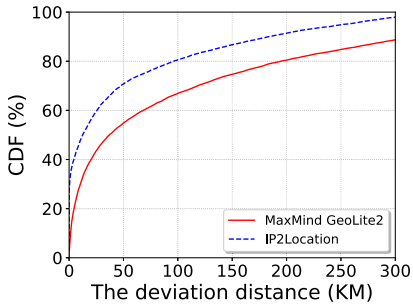


Fig. 10. Compared with the commercial geolocation database.

based on webcam/open-source landmarks, (2) CBG [8] based on webcam/open-source landmarks, (3) Octant [9] based on webcam/open-source landmarks, and (4) Spotter [10] based on webcam/open-source landmarks. Similarly, all the algorithms use the last shared routers with the shortest ping to measure the RTT. We use multilateration to narrow down the region and pinpoint the target to the node with the smallest relative latency.

Aforementioned, all geolocation algorithms need to use the correlation between distance and time latency for geolocation estimation. The difference between real distance and estimated distance directly determines the degree of geolocation accuracy for a target. The lower the distance difference, the higher the geolocation accuracy. In geolocation algorithms, a small relative time usually requires geographic proximity to the target. Figure 11 shows the calibration of relative delay and distance, where a blue point represents a pair of a landmark and a target. We randomly select 80 pairs of landmarks and targets from our dataset. The X-axis is the relative-delay and the Y-axis is the physical geographic distance between a landmark and a target. The dashed line is the baseline for the distance-delay relationship. When the relative delay is large, the estimated distance errors become large under the fixed empirical speed. If we could find a small relative delay for the target, the geolocation algorithms can accurately predict the target location. The superiority of webcam landmarks is to provide the worldwide coverage and have a high probability of finding at least one landmark close to the target.

We observe the significant improvements of GeoCAM algorithm in geolocation accuracy under webcam landmarks. Figure 12 illustrates the CDF of the geolocation errors on the residential hosts for evaluating the performance of GeoCAM. The X-axis is the error distance (kilometer) between the ground truth and the estimated geolocation. We observe that GeoCAM using webcam landmarks achieves higher accuracy than using open-source landmarks. When GeoCAM uses webcam landmarks, 20% of targets have less 89KM errors and 40% of targets have less 258KM errors. By contrast, when

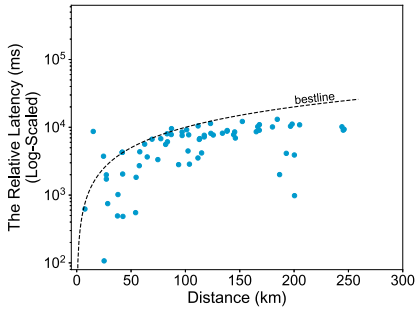


Fig. 11. Calibration scatter plots of the relative-delay-distance.

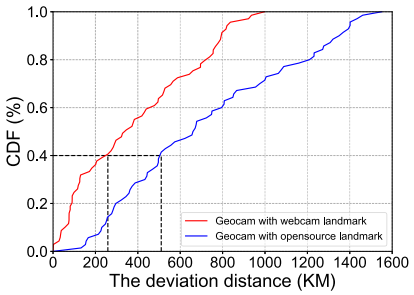


Fig. 12. GeoCAM with webcam/open-source landmarks.

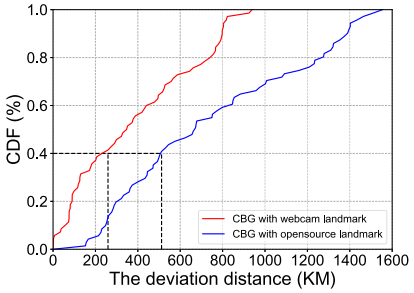


Fig. 13. CBG with webcam/open-source landmarks.

GeoCAM uses open-source landmarks, 20% of targets are less 321KM errors and 40% of targets have less 510KM errors. GeoCAM achieves a 60% higher accuracy when algorithms adopt webcam landmarks.

Next, we observe the significant gains of other three algorithms using webcam landmarks, in comparison with using open-source landmarks. Figure 13 plots the performance of CBG with webcam landmarks and open-source landmarks, and Figure 14 depicts the performance of Octant. We observe that these two algorithms perform similarly in geolocation accuracy. Using open-source landmarks, both CBG and Octant are close to having 40% of hosts with less 510KM errors, and 60% of hosts with less than 800KM errors. Their geolocation accuracies significantly increase when both use webcam landmarks, having 60% of hosts with less 440KM errors. Figure 15 shows the CDF of the geolocation errors on the residential hosts for Spotter using webcam landmarks, where 20% of targets have 205KM errors and 40% of targets have 334KM errors. We observe that Spotter has the worst performance compared with the other three algorithms. It might be that Spotter underestimates the speed packets can travel, leading to inaccurate prediction regions. Overall, webcam landmarks significantly improve the geolocation accuracy for all four geolocation algorithms (GeoCAM, CBG, Octant, and Spotter) in comparison with existing landmarks (i.e., open-source landmarks).

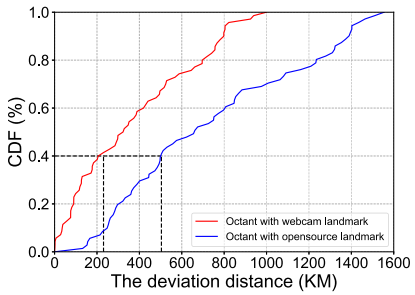


Fig. 14. Octant with webcam/open-source landmarks.

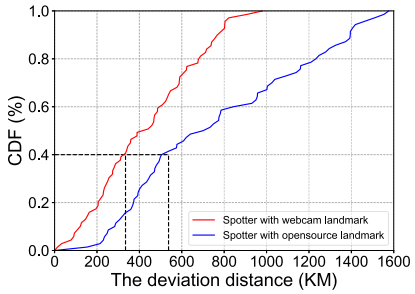


Fig. 15. Spotter with webcam/open-source landmarks.

TABLE XII
THE IMPACT OF THE VP NUMBER ON THE GEOLOCATION SERVICE

VP Number	6	8	10
Average Qualified Traceroutes	2.1	2.6	3.2
GeoCAM Performance (top 20%)	127KM	95KM	89KM

C. VP Impact on Performance

For the geolocation algorithms based on active measurements, they rely on the quality and quantity of VPs, and our work is no exception. However, the focus of our work is not on investigating the impact of VPs on geolocation performance. Thus, in this study, we only use 10 VPs across the U.S. and Europe to actively measure traces for geolocation services.

The quality of a VP varies with respect to different targeted IPs. Dependent upon a targeted IP, sometimes a VP can obtain a high-quality traceroute, sometimes it cannot. When a VP is close to a target, the RTT will be low, and the error distance will be small too. However, when a traceroute's quality is poor, the latency will be high (hundred milliseconds), and it will not be capable of providing precise measurements to geolocate the corresponding IP address. In such scenarios, timeouts often occur when VPs send the traceroute requests to IP addresses. In our experiments, after removing those unqualified VPs, the average number of qualified traceroutes (among 10 VPs) is only 3.2 for an IP address. Note that the geolocation accuracy would be further improved if we leveraged more high-quality VPs to measure latency and route of IP targets. Table XII lists the impact of varying VP numbers upon the geolocation service.

VII. DISCUSSION

Our study shows that GeoCAM makes a first step towards fully automated landmark generation based on webcam devices. On the other hand, our current design could be further enhanced. Here, we discuss the limitations and concerns of GeoCAM.

Coverage Limitation: In the data collection, we crawled 100 different websites and retrieved 1.9 million webpages. However, we acknowledge that GeoCAM cannot exhaustively gather all live webcams on the Internet. Besides those 100 websites, some less popular websites (e.g., blogs or forums) might also distribute webcams to the public. In our future work, we plan to crawl more websites to find a even larger number of webcams for being used as landmarks.

Privacy Concerns: Since we generate landmarks based on webcams, user privacy would be a concern. However, we only scrape webpages from websites and extract location information of candidate landmarks, which are already publicly accessible on the Internet. If a landmark violates user privacy or permission, we can directly remove it from the candidate set. More importantly, for those webcams used as landmarks, we never attempt to log into or control them.

Webcam Image Recognition: As aforementioned, we used the Tesseract [23] to recognize the text from a webcam image. Here, we only focus on embedded characters on the webcam image (geographic names and timestamps). With only a few exceptions, geographic names derived from webcam images are already included in the geolocation names from HTML elements. Overall, only a small portion of texts extracted from images are unique and most of them are timestamps.

Webcam Validation: is an important but difficult issue to handle, because landmarks (both open-source platforms and ours) could suffer from the untruthfulness of geolocation information, where only owners can correct the misinformation. Webcam devices are submitted by device owners or volunteers, coupling with explicit IP addresses. In those cases, the geolocation information tied with a webcam is normally true. In addition, we propose a manual way to validate the geolocation information of a webcam. For webpages with specific geolocation information, we can manually narrow the search scope. For webpages without geolocation information, they usually have geographic coordinates, or they will not appear in our system. Then we can check online map and compare snapshots with street views to validate the geolocation information.

Impacts of CDNs: Similarly, CDNs could affect the performance of webcam landmarks. We manually labeled 200 randomly selected landmarks in our dataset. We found there are only 8.36% of webcams that either share an IP but run at different ports or run in a cloud, which is small. The reason is that webcam owners are prone to host live webcam streams on individual IP addresses, as shown in Figure 1. In addition, we will filter out unqualified webcam landmarks in the future work. For instance, we can use a CDN list to filter out those IP addresses behind a CDN.

Landmark Error: is an inevitable issue even when we automatically mine and collect landmarks from online sources. If an error occurs, there are two ways to address it: (1) a validation through crowdsourcing, e.g., [32] and (2) manual calibration by searching webcam images with the description in search engines. So far, how to automatically remove incorrect landmarks is an unsolved problem. We will address the removal of incorrect landmarks in our future work and update calibrated landmarks on our website [11].

VIII. RELATED WORK

IP-based geolocation has two categories: client-dependent and client-independent. The client-dependent approach relies on the client-side support, such as GPS and WiFi signals.

GeoCAM is a client-independent approach that does not require any client-side support. In this section, we survey the previous client-independent works.

Data Mining-Based Approach: Given geographical information in public webpages/datasets, this approach derives IP-location relationships using data mining techniques. Liu *et al.* [33] utilized check-in data from social networks to generate IP-user-location relationships. They mined check-in patterns from login logs and inferred users' geolocation and IP addresses (such as *home*, *office*, and *others*). Such a method requires private login logs and only works for active users. Padmanabhan *et al.* [34] proposed geographical clusters based on address prefixes in the border gateway protocol (BGP) tables, and pinpointed a host using the landmarks within the same cluster. Huffaker *et al.* [35] leveraged geographical hints from hostnames to geolocate a large set of routers on the Internet, but geo-hints of hostnames are limited and probably inconsistent with actual locations.

Network Measurement-Based Approach: This method uses latencies and topologies among VPs and targets to estimate the geo-location of a target host. Gueye *et al.* [8] proposed a constraint-based geolocation (CBG), which uses geographical distance and multilateration to locate a target host. CBG utilizes the bestline estimation to reduce errors induced by inflated latencies and indirect routers. Katz-Bassett *et al.* [36] proposed a topology-based geolocation (TBG), which introduces network topology constraints for improving the performance of CBG. Wong *et al.* [9] proposed a generic framework called Octant that locates IP hosts by incorporating latency, network topology, and hostnames. Laki *et al.* [10] proposed Spotter to use the highest probability density to geolocate target hosts.

Geolocation Database: There are several available IP geolocation databases, including MaxMind GeoLite2 [37], GeoIP2 [38], IP2Location [31], and NetAcuity [39]. Those databases provide physical locations to IP addresses at the country-level and city-level granularities. Prior works [2], [40], [41] have evaluated the accuracy of databases from endpoints to router geolocations. Poese *et al.* [40] found that the number of unique geographical locations are much less than the number of IP blocks, which cannot build accurate mappings between physical locations and IP hosts. Shavitt *et al.* [41] demonstrated that most of databases cannot archive the accuracy they acclaimed at the country level, and perform poorly at the city level. Gharaibeh *et al.* [2] analyzed router geolocation accuracy across those databases, and found that databases are not reliable for geolocating routers at the city level.

Landmark: The accuracy of IP-based geolocation is heavily dependent on the density of landmarks. There are several platforms for providing landmarks with known IP addresses and accurate geolocations to the public. PlanetLab [5] is a geographically distributed network testbed, running 1,353 nodes at 717 sites, most of which are universities or research institutions. As yet, we find that only 420 nodes are still available. PerfSONAR [3] is a network measurement tookit designed to provide federated coverage of paths, and help to establish end-to-end usage expectations. PingER [4] is a monitoring infrastructure to understand network performance and allocate resources to optimize performance between laboratories. Ciavarrini *et al.* [42] proposed a crowdsourcing method to use smartphones as vantage points for geolocating target hosts. Jia *et al.* [43] employed adjacent IoT devices to verify the geographic location of cloud hosts. They leveraged IoT devices as landmarks and proposed a vote-based closest-shortest mechanism to geolocate cloud hosts.

Mining web services has been proposed to increase the number of landmarks. Guo *et al.* [6] and Li *et al.* [44] extracted geographical information from webpages and built the relationships between IP addresses of websites and their physical locations. However, the location information extracted from websites is not the actual location of websites. Those landmarks belong to the /24 subnet, only at the city-level granularity. Wang *et al.* [7] also leveraged geographical information on websites and searched them through online maps for generating landmarks. Unfortunately, for those landmark generation approaches based on mining web services, the mappings among domain names, IP addresses, and locations are not stable, due to the widely used cloud services and CDN. Thus, their landmarks have unstable and unverifiable issues, resulting in unavailability of landmarks. By contrast, we generate landmarks based on live webcams that are stable with time and can be verified by manual inspection on their live streams. Moreover, GeoCAM generates landmarks at the latitude/longitude granularity.

IX. Conclusion

IP-based geolocation heavily relies on the number of high-quality landmarks. As the pervasiveness of IoT systems, an increasing number of webcams connected to the Internet have become an ideal set of candidates for being used as landmarks. In this paper, we proposed a framework GeoCAM to automatically generate webcam landmarks and provide IP-based geolocation services. Specifically, GeoCAM periodically monitors those websites that host live webcams and uses the natural language processing technique to extract the IP addresses and latitudes/longitudes of webcams. We demonstrated the viability of different geolocation algorithms, including CBG, Octant, Spotter, and our hybrid algorithms. We conducted experiments to evaluate the performance and effectiveness of GeoCAM. Our results show that GeoCAM automatically detects webcams with 94.2% precision and 90.4% recall, and can generate 16,863 stable and fine-grained landmarks. These webcam landmarks are two orders of magnitude more than the landmarks used in existing geolocation services. Therefore, GeoCAM is capable of providing a geolocation service with high accuracy and wide coverage.

Acknowledgment

The authors are grateful to our anonymous reviewers for their insightful feedback.

References

[1] J. A. Muir and P. C. V. Oorschot, "Internet geolocation: Evasion and counterevasion," *ACM Comput. Surv.*, vol. 42, no. 1, pp. 4:1–4:23, Dec. 2009.

[2] M. Gharaibeh, A. Shah, B. Huffaker, H. Zhang, R. Ensafi, and C. Papadopoulos, "A look at router geolocation in public and commercial databases," in *Proc. ACM Internet Meas. Conf.*, Nov. 2017, pp. 463–469.

[3] A. Hanemann *et al.*, "PerfSONAR: A service oriented architecture for multi-domain network monitoring," in *Proc. Int. Conf. Service-Oriented Comput. (ICSOC)*, 2005, pp. 241–254.

[4] W. Matthews and L. Cottrell, "The PingER project: Active Internet performance monitoring for the HENP community," *IEEE Commun. Mag.*, vol. 38, no. 5, pp. 130–136, May 2000.

[5] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the Internet," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 59–64, Jan. 2003.

[6] C. Guo, Y. Liu, W. Shen, H. J. Wang, Q. Yu, and Y. Zhang, "Mining the Web and the Internet for accurate IP address geolocations," in *Proc. IEEE 28th Conf. Comput. Commun. (INFOCOM)*, Apr. 2009, pp. 2841–2845.

[7] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang, "Towards street-level client-independent IP geolocation," in *Proc. 8th USENIX Conf. Networked Syst. Design Implement. (NSDI)*. Baltimore, MD, USA: USENIX Association, 2011, pp. 365–379.

[8] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of Internet hosts," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1219–1232, Dec. 2006.

[9] B. Wong, I. Stoyanov, and E. G. Sirer, "Octant: A comprehensive framework for the geolocalization of Internet hosts," in *Proc. 4th USENIX Conf. Networked Syst. Design Implement. (NSDI)*. Baltimore, MD, USA: USENIX Association, 2007, p. 23.

[10] S. Laki, P. Matray, P. Haga, T. Sebok, I. Csabai, and G. Vattay, "Spotter: A model based active geolocation service," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2011, pp. 3173–3181.

[11] S. Wang. (2021). *The Landmark Dataset for GeoCAM*. [Online]. Available: http://www.infolab.top/research/GeoCAM/

[12] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering Internet-of-Things devices," in *Proc. 27th USENIX Secur. Symp. (USENIX Security)*. Baltimore, MD, USA: USENIX Association, 2018, pp. 327–341. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/feng

[13] K. Yang, Q. Li, and L. Sun, "Towards automatic fingerprinting of IoT devices in the cyberspace," *Comput. Netw.*, vol. 148, pp. 318–327, Jan. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618306856

[14] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, Aug. 2007.

[15] J. Song, Q. Li, H. Wang, and L. Sun, "Under the concealing surface: Detecting and understanding live Webcams in the wild," in *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 1, pp. 1–25, Mar. 2020.

[16] JPEG. (1992). *Joint Photographic Experts Group for Lossy Compression for Digital Images*. [Online]. Available: https://jpeg.org/about.html

[17] VLC. (2001). *Media Player Originated by the VideoLan Software*. [Online]. Available: https://www.videolan.org/

[18] FFmpeg. (2000). *A Complete, Cross-Platform Solution to Record, Convert and Stream Audio and Video*. [Online]. Available: https://www.ffmpeg.org/

[19] (2010). *Common Crawl*. Accessed: Apr. 10, 2019. [Online]. Available: https://commoncrawl.org/

[20] (2004). *Beautiful Soup*. Accessed: Apr. 10, 2019. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/

[21] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[22] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol. (NAACL)*, 2003, pp. 173–180.

[23] (1984). *Tesseract Open Source OCR Engine*. Accessed: Apr. 10, 2019. [Online]. Available: https://github.com/tesseract-ocr/tesseract

[24] (2004). *Openstreetmap Wiki*. Accessed: Apr. 10, 2019. [Online]. Available: https://wiki.openstreetmap.org/w/index.php

[25] Z. Hu, J. Heidemann, and Y. Pradkin, "Towards geolocation of millions of IP addresses," in *Proc. ACM Conf. Internet Meas. Conf. (IMC)*, 2012, pp. 123–130.

[26] A. Ziviani, S. Fdida, J. F. de Rezende, and O. C. M. B. Duarte, "Improving the accuracy of measurement-based geographic location of Internet hosts," *Comput. Netw.*, vol. 47, no. 4, pp. 503–523, Mar. 2005.

[27] (1987). *Traceroute*. Accessed: Apr. 10, 2019. [Online]. Available: http://traceroute.sourceforge.net/

[28] (1999). *BGP Routing Table Analysis*. Accessed: Apr. 10, 2019. [Online]. Available: http://thyme.apnic.net/

[29] (2016). *Ripe Atlas*. Accessed: Apr. 10, 2019. [Online]. Available: https://atlas.ripe.net/

[30] G. 2.0. (2020). *Locate the Coordinates of Addresses, Cities, Countries, and Landmarks Across the Globe Using Third-Party Geocoders and Other Data Sources*. [Online]. Available: https://geopy.readthedocs.io/en/latest

[31] (2001). *IP2location*. [Online]. Available: https://www.ip2location.com/ Accessed: Apr. 10, 2019.

[32] P.-Y.-P. Chi, M. Long, A. Gaur, A. Deora, A. Batra, and D. Luong, "Crowdsourcing images for global diversity," in *Proc. 21st Int. Conf. Hum.-Comput. Interact. Mobile Devices Services (MobileHCI)*. New York, NY, USA: Association for Computing Machinery, Oct. 2019, doi: 10.1145/3338286.3347546.

[33] H. Liu, Y. Zhang, Y. Zhou, D. Zhang, X. Fu, and K. K. Ramakrishnan, "Mining checkins from location-sharing services for client-independent IP geolocation," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 619–627.

[34] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for Internet hosts," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, 2001, pp. 173–185.

[35] B. Huffaker, M. Fomenkov, and K. Claffy, "DRoP: DNS-based router positioning," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 5–13, Jul. 2014.

[36] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards IP geolocation using delay and topology measurements," in *Proc. 6th ACM SIGCOMM Internet Meas. (IMC)*, 2006, pp. 71–84.

[37] (2002). *Maxmind Geolite2 Free Downloadable Databases*. Accessed: Apr. 10, 2019. [Online]. Available: https://dev.maxmind.com/geoip/geoip2/geolite2/

[38] (2002). *Maxmind Geoip2 Database Maxmind*. Accessed: Apr. 10, 2019. [Online]. Available: https://www.maxmind.com/en/geoip2-databases/

[39] (1999). *Netacuity*. Accessed: Apr. 10, 2019. [Online]. Available: https://www.digitalelement.com/solutions/

[40] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "IP geolocation databases: Unreliable?" *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 53–56, Apr. 2011.

[41] Y. Shavitt and N. Zilberman, "A geolocation databases study," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 10, pp. 2044–2056, Dec. 2011.

[42] G. Ciavarrini, V. Luconi, and A. Vecchio, "Smartphone-based geolocation of Internet hosts," *Comput. Netw.*, vol. 116, pp. 22–32, Apr. 2017.

[43] D. Jia, L. Liu, S. Jia, and J. Lin, "VoteGeo: An IoT-based voting approach to verify the geographic location of cloud hosts," in *Proc. IEEE 38th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Oct. 2019, pp. 1–9.

[44] D. Li *et al.*, "IP-geolocation mapping for moderately connected Internet regions," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 2, pp. 381–391, Feb. 2013.

[45] Z. Wang, Q. Li, J. Song, H. Wang, and L. Sun, "Towards IP-based geolocation via fine-grained and stable webcam landmarks," in *Proc. Web Conf.*, Apr. 2020, pp. 1422–1432, doi: 10.1145/3366423.3380216.

Qiang Li received the Ph.D. degree in computer science from the University of Chinese Academy of Sciences in 2015. He is currently an Associate Professor with the School of Computer and Information Technology, Beijing Jiaotong University, China. His research interests include the Internet of Things, networking systems, network measurement, machine learning for cyber-security, and mobile computing.

Zhihao Wang received the B.S. degree from the University of Science and Technology of China in 2015. He is currently pursuing the Ph.D. degree with the University of Chinese Academy of Sciences, China. His research interests include cyber security, the Internet of Things security, and Internet measurement.

Dawei Tan received the B.E. degree from Beijing Jiaotong University, China, in 2018, where he is currently pursuing the M.S. degree. His research interests include cyber security and Internet measurement.

Jinke Song received the master's degree in computer science from Beijing Jiaotong University, China, in 2017, where he is currently pursuing the Ph.D. degree with in the School of Computer and Information Technology. His main research interests include cyberspace security and the Internet of Things security.

Haining Wang (Fellow, IEEE) received the Ph.D. degree in computer science and engineering from the University of Michigan, Ann Arbor, MI, USA, in 2003. He is currently a Professor with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA. His current research interests include security, networking systems, and cloud computing.

Limin Sun received the Ph.D. degree from the National University of Defense Technology in 1998. He is currently a Professor with the Institute of Information Engineering—Chinese Academy of Sciences (CAS). His research interests include the security of the Internet of Things, mobile vehicle networks, wireless sensor networks, and mobile IP. He is an Editor of the *Journal of Computer Science* and *Journal of Computer Applications*.

Jiqiang Liu (Senior Member, IEEE) received the B.S. and Ph.D. degrees from Beijing Normal University, in 1994 and 1999, respectively. He is currently a Professor with the School of Computer and Information Technology, Beijing Jiaotong University. His current research interests include cryptographic protocols, privacy preserving, and network security.