OSRM-CCTV: Open-source CCTV-aware routing and navigation system for privacy, anonymity and safety (Preprint)

Lauri Sintonen, Hannu Turtiainen, Andrei Costin, Timo Hämäläinen and Tuomo Lahtinen

Abstract—For the last several decades, the increased, widespread, unwarranted, and unaccountable use of Closed-Circuit TeleVision (CCTV) cameras globally has raised concerns about privacy risks. Additional recent features of many CCTV cameras, such as Internet of Things (IoT) connectivity and Artificial Intelligence (AI)-based facial recognition, only increase concerns among privacy advocates. Therefore, on par CCTV-aware solutions must exist that provide privacy, safety, and cybersecurity features. We argue that an important step forward is to develop solutions addressing privacy concerns via routing and navigation systems (e.g., OpenStreetMap, Google Maps) that provide both privacy and safety options for areas where cameras are known to be present. However, at present no routing and navigation system, whether online or offline, provide corresponding CCTV-aware functionality.

In this paper we introduce OSRM-CCTV – the first and only CCTV-aware routing and navigation system designed and built for privacy, anonymity and safety applications. We validate and demonstrate the effectiveness and viability of the system on a handful of synthetic and real-world examples. To help validate our work as well as to further encourage the development and wide adoption of the system, we release OSRM-CCTV as open-source.

Index Terms—Privacy-enhancing technologies and anonymity, Usable security and privacy, Research on surveillance and censorship, Privacy, Anonymity, Surveillance, Safety, Routing, Navigation, Mapping, OSM, OSRM, open-source.

I. INTRODUCTION

In the modern world, public spaces of many cities are being surveilled by close-circuit television (CCTV) cameras to a considerable extent. It is estimated that globally there are around 770 million CCTV cameras in use and that their amount could rise up to one billion during the year 2021 [6]. As an example, there are approximately half a million CCTV cameras in London and an average person living there is recorded on camera 300 times everyday [3]. In the United States, it is likely that people get recorded by a CCTV camera over fifty times per day [4]. In 2019, a person documented 49 CCTV cameras on the way to work in New York City [5] and described it dystopian.

The discourse on CCTV surveillance has ethical dimensions. Von Hirsch argues that CCTV surveillance is sometimes covert and that often people believe that they are not under CCTV surveillance when they actually are [6]. Furthermore, according to a 2016 survey, an average citizen of the United States assumed being recorded by four or less CCTV cameras per day while the real figure is likely over 10 times larger [4]. Considering the amount of CCTV cameras having been installed globally and the fact that people can be detected and recorded by them, adding face recognition to the pattern opens up an unsettling possibility to also automatically identify people by CCTV cameras [7], [8], [9]. Overall, there is controversy about the ethics of CCTV surveillance and whole books have been written on the subject [6], [10]. Moreover, the CCTV cameras, Digital Video Recorders (DVR) and Video Surveillance Systems (VSS) are notoriously known to be vulnerable to cybersecurity attacks and hacks [11], therefore it is reasonable to assume that the CCTV cameras overlooking public places and routes may be at any times under the control of unauthorized persons hence posing a direct threat to privacy.

In this context, we argue that it is important to create CCTV-aware solutions and technologies that allow people at least a slight option to choose between being under CCTV surveillance (or not!) whenever they walk, cycle or drive in public spaces. We approach the question from the perspective of routing and navigation. While there is a substantial amount of existing routing solutions and technologies [12], [13], [14], [15], [16], [17], to the best of our knowledge none of them focuses on the issue at hand, to provide a CCTV-aware routing solution.

In this paper, we propose one such CCTV-aware solution. Our CCTV-aware solution enables two modes of routing. Firstly, privacy-mode which aids maintaining privacy by choosing a route where CCTV cameras are avoided. Privacy-mode is strict in the sense that it will never choose a route that leads through the field of vision of a CCTV camera. Avoiding CCTV cameras would be desirable anytime when privacy or anonymity is important. Secondly, safety-mode which aids maintaining safety by choosing a weighed route that leads through the fields of vision of CCTV cameras. Safety-mode is called safe because sometimes (e.g., at night and for the preference of staying physically safe) one would prefer to be actually detected and recorded by CCTV cameras.

It is worth to mention that neither of these routing modes are guaranteed to completely maintain either privacy or safety as described in this paper. This is mainly because of the lack of up-to-date and accurate data about the CCTV cameras. The relevant data in question would contain information about for example CCTV camera model and lense, the radius, the angle...
of view and direction of the CCTV cameras, and data like this is more often than not is unavailable. Currently, to our knowledge there is no automated way to gather data about CCTV cameras to be used in our CCTV-aware approach. However, regarding the collection of relevant data we have also developed and demonstrated the first Computer Vision (CV) models that can be used to detect CCTV cameras in images (e.g., street-level/street view, indoors). For example, our CV models achieve an accuracy of up to 98.7\% \cite{18} and we are currently in the process of mapping large areas around the globe.

The rest of the paper is organized as follows. In Section\ II we present earlier work that has been done to allow different types of routing. Then, in Section\ III we provide an outline of how we have implemented our CCTV-aware solution. Next, in Section \ IV we describe how our CCTV-aware solution functions where we present different examples of both privacy-mode and security-mode on a synthetic evaluation map where different routing scenarios can be demonstrated. We present examples based on real data in Section \ V. We conclude this paper with Section \ VI.

Our main contributions with this work are:

1) We are the first (to the best of our knowledge) to propose, implement, and evaluate CCTV-aware routing for privacy, anonymity and safety applications.
2) We demonstrate the usefulness and the feasibility of CCTV-aware routing applied to both synthetic and real-world examples.
3) We release (upon peer-review acceptance) the relevant artifacts (e.g., code, data, documentation) as open-source: https://github.com/Fuziih

II. RELATED WORK

OpenStreetMap (OSM) is a collaborative open-source project, which aims to supply free and editable maps of the world. Many reputable projects \cite{12, 13, 14, 15, 16, 17} (e.g., routing solutions and technologies for different use-cases) base their work on the OSM. Most routing solutions focus on standard driving, cycling, and walking schemes. In addition to this, there are some less common/popular yet useful solutions, such as finding wheelchair accessible routes \cite{19, 20, 21}, finding curvy roads \cite{22}, and solutions that calculate routes based on the day arc \cite{23, 24, 25}.

Recently, Sirtiiraya et al. \cite{26} argued that in the modern contemporary society, pedestrians prefer to set “alternative routing criteria” rather than the more common routing criteria such as duration and length. The authors refer to this as “routing based on different qualities”, i.e., qualitative-aware routing. The authors identified five routing quality categories: safety, well-being, effort, exploration, and pleasure. Correspondingly to our work, safety category is acknowledged by the authors as the need for avoiding areas with a high occurrence of past criminal activities. Although privacy is not mentioned nor categorized, it could be accommodated into the safety category of \cite{26}.

When pursuing to achieve a CCTV-aware routing solution, inspiration could be drawn from the day arc routing solutions. For example, Olaverri Monreal et al. \cite{23} created sun-avoiding routing solution and Ma \cite{24, 27} developed a solution to either avoid or face the sun during travel. Recently, Deilami et al. \cite{25} provided a routing solution, which takes tree shading into consideration. The routing solutions in these works could be used to establish our CCTV-aware routing, however, there are a few pitfalls with that approach. Firstly, the data requirements for these solutions are quite specific and reaching them could prove to be problematic. Secondly, as an OSM “Way” object is defined as a collection of nodes, a line, and the width is saved only as a key–value tag \cite{28, 29}, this fundamental issue would need to be addressed.

Privacy and safety routing has been previously researched to a certain, yet limited, degree. Bao et al. \cite{30} proposed a routing solution with multiple safe pedestrian routing conditions. Hirozaku et al. \cite{31} took into account street lights for safer walking routes. Keler and Mazimpaka \cite{32} employed crowd-sourcing in their efforts to create safe routing. Lastly, Tessio et al. \cite{33} were set on finding ways for users to find routing through green areas, social places and quieter streets. Lahtinen et al. \cite{34} investigated just the initial feasibility of CCTV-aware routing solutions and the quantitative impact on the routing options.

However, none of the previous works address the privacy, anonymity and safety routing approaches given the exponential growth of CCTV camera and video surveillance presence in public spaces and routes \cite{5}. With this work we try to close this gap as it is also part of our bigger vision revolving around CCTV cameras, video surveillance, digital privacy and anonymity. Therefore, we present that related work as well. First, we designed a Computer Vision (CV) model for detecting CCTV and video surveillance cameras in images and video frames \cite{18}. Second, in order to improve and accelerate the training of our CV model \cite{18}, we created from scratch a novel image annotation tool \textit{BRIMA} \cite{35} which is a browser extension that provides an effortless way to map CCTV cameras using services such as Google Maps Street View \cite{36}, Yandex \cite{37} or Baidu \cite{38}. Third, we published an initial feasibility-study work where we mapped 450 CCTV cameras in the city of Jyväskylä, Finland, and conducted routing experiments concluding that our system is feasible for further exploration and development, as it supports both the privacy-first and safety-first approaches \cite{39}.

III. IMPLEMENTATION DETAILS

This section contains a description of the most relevant tools and libraries as well as the data manipulation process that are used to enable CCTV-aware routing with two alternative modes called privacy-mode and safety-mode. The routing process is two-fold. First, OSM data is processed so that the resulting OSM file contains 1) the CCTV cameras as nodes, 2) entrance and exit nodes of the areas that intersect with the fields of vision of CCTV cameras, and 3) ways that are not merely lines without a width but ones that have an actual width on the map. OSM Way objects do not have an actual width as they are defined as a collection of nodes, and the width is saved only as a key–value tag \cite{28, 29}. Having
an actual width is essential in enabling CCTV-aware routing, because if a CCTV camera surveys only part of one side of a way, one should still be able to traverse the other side of that way, and width is needed to allow this possibility. Second, our OSM-CCTV routing implementation itself is based on Open Source Routing Machine (OSRM) [40, 41] using a custom OSM file created in the processing phase. Privacy-mode and safety-mode are described in Lua [42] programming language as customized profiles that the standard OSRM supports by default [43].

A. Tools and Libraries

In this section, the most relevant tools and libraries used for data processing are described. The effort enabling CCTV-aware routing is heavily weighed on processing OSM data and the programming language used for that is Python [44]. The relevant tools and libraries are Osmium [45] and its Python bindings, PyOsmium [46], Shapely [47] and PyProj [48]. We run the backend using Docker and Docker-Compose [49] on an Ubuntu 18.04 [50] platform. Internal routing between Docker containers is performed with Nginx [51].

Osmium is a “multipurpose command line tool for working with OpenStreetMap data based on the Osmium library” [45]. In our work, PyOsmium [46], the Python bindings of the Osmium library, is the main library used in adding cameras and splitting ways in OSM data, more specifically, OSM XML data [52]. PyOsmium is also used to write the resulting OSM file that is used in routing with Open Source Routing Machine. Shapely is a “Python package for manipulation and analysis of planar geometric objects” [47]. It is used to transform OSM data to a more modifiable form to perform most of the necessary geometric transformations and calculations that are necessary in manipulating OSM data. Since Shapely assumes data on a Cartesian plane, we use PyProj [48] for spatial projections.

B. Data Processing

In this section, the data processing phase of our CCTV-aware routing solution is described. This involves the necessary steps in order to enable CCTV-aware routing with OSRM-CCTV, and can be divided largely into two parts:

1) managing OSM file to contain CCTV data
2) preparing final OSM file for routing with OSM-CCTV

1) Manipulating OSM data: Algorithm 1 describes the most important aspects of the OSM data manipulation phase in pseudocode. It is simplified and not a completely accurate representation of everything that happens in the code. In essence, the algorithm uses an existing OSM XML file [52] and writes a new one based on a Comma-Separated Value (CSV) file which includes the properties of the cameras that are to be added to the resulting OSM XML file. These properties are GPS coordinates (e.g., latitude, longitude), type, radius, angle and direction of vision of each new camera. Two types for the cameras are defined: directed and round. A round camera has a 360 degree angle of view, while an angle narrower than that would represent a directed camera type. Radius means how far

```
Algorithm 1 OSM data manipulation.

1: procedure MAIN(data):
2:   for all line in data do
3:     coords = line["location of camera"]
4:     cam_type = line["directed"] OR line["round"]
5:     radius = line["how far the camera sees"]
6:     angle = line["how wide the camera sees"]
7:     direction = line["camera pointing direction"]
8:     call handle_surveillance_area(coords, cam_type, radius, angle, direction, writer, osm)
9:   end for
10: compile_data_and_add_new_and_old_ways_to_new_osm
11: end procedure

12: procedure HANDLE_SURVEILLANCE_AREA(coords, cam_type, radius, angle, direction, writer, osm):
13:   writer.add_camera(coords)
14:   determine_surveillance_area(cam_type)
15:   extract_area = determine_extract_area(coords)
16:   call create_tmp_extract(extract_area, osm_file)
17:   for all travellable way in tmp_extract.ways do
18:     add_splits
19:     gather_data_for_compilation
20:     writer.add_way_nodes
21:     call handle_intersections_aov_way
22:     gather_data_for_compilation
23:   end for
24: end procedure

25: procedure CREATE_TMP_EXTRACT(extract_area, osm_file):
26:   save_tmp_extract_on_disk(from osm_file)
27:   split_ways_in_tmp_extract -> use split_ways.py
28:   for all way in travellable_ways do
29:     create_parallel_ways
30:   end for
31: end procedure

32: procedure HANDLE_INTERSECTIONS_AOV_WAY:
33:   if intersects then
34:     add_entrance_and_exit_nodes_to_cam_aov_boundary
35:     create_weights_for_safety_mode
36:     gather_data_for_compilation
37:   end if
38:   gather_data_of_split_ways_for_compilation
39:   gather_data_of_regular_ways_for_compilation
40: end procedure
```
the camera can “see” for its particular function, for example, “detect faces”, “recognize faces”, or “recognize license-plate numbers” (i.e., depending on privacy/safety-impact scenario being modeled and evaluated). The radius can be chosen according to case-specific needs (e.g., depending on characteristics of CCTV cameras and lenses), and its exact value is not relevant for performance evaluations in this paper. Angle, or horizontal Field of View (FoV) in our case, describes how wide the angle of the camera vision is, hence, it also affects the determination of the type of the camera between round and directed. Direction means the direction to which the center of the field of vision of the camera is pointing.

The resulting OSM XML file will include the new CCTV cameras as nodes. Also, the intersection points of routable ways and the fields of vision of the CCTV cameras are represented as nodes on the ways. These nodes are marked with an access tag that points to surveillance. From the perspective of CCTV-aware routing, the nodes with an access tag are enough in determining when one would enter the field of vision of a CCTV camera. Depending on the selected routing mode, the router would either avoid the access tags that point towards surveillance or pursue routing through these nodes. Finally, the ways in the resulting OSM XML are split in three whenever there is a CCTV camera nearby. The three split ways represent the left, the middle and the right side of the way.

Next, some clarification is provided for some parts in the pseudocode Algorithm 1. In PyOsium [46], the Python bindings of the Osmium library [45], writer is used for writing OSM objects on the resulting OSM file. Hence, when writer is mentioned, something is written on that file. The osm parameter seen for instance on line 12 references to the original OSM file that has no added CCTV cameras.

Based on the type, the angle and the radius of the field of vision of the CCTV camera, the subroutine determine_surveillance_area is used for determining the area that the camera surveys. When the camera type is “round” and the angle of the camera vision is 360 degrees, the surveilled area is circular. In the case of a directed CCTV camera, the surveilled area is a sector of a circle. In this subroutine, determine_extract_area determines the area which is extracted from the original OSM file. This area is a circle with a diameter of 50. In create_temp_extract, the determined extract area is used to execute the extraction and writing a temporary OSM file on the disk. Also, all traversable ways that are found inside the circle are split in two using another script, split_ways.py. Splitting ways means creating two additional ways parallel to the original way. The distance of each of the parallel ways will be the value of the width tag of the original way divided by 2 – creating a left and a right border of the way. A final temporary file is created. Data is manipulated in intermediary extracts to mitigate the memory-extensiveness of manipulating large OSM XML files.

Now, reaching the line 17 in Algorithm 1 all ways of possible travel in the newly-created temporary OSM file are iterated. The split ways are added to the the resulting OSM file and the intersections with the ways and the borders of the CCTV surveillance areas that were determined earlier are handled in handle_intersections_aov_way. When a way penetrates a camera-surveilled area, one or two intersections occur. Entrance and exit nodes are added at the intersection points. Also, weights for OSM-CCTV traffic updates [53] used with safety-mode are added and written on disk to a CSV file.

Finally, in Algorithm 1 there are mentions about gathering data. These mentions refer to every time that some data is saved into variables for the final compilation executed in the end of Algorithm 1 on line 10, at compile_data and add_new_and_old_ways_to_new_osm.

2) Preparing data for routing with OSM-CCTV: The second part of the data processing is executed by OSM-CCTV, which involves parts for extracting, partitioning and customizing both the OSM file (resulting from the first part of the data processing phase) and the profile file which determines which mode is used – privacy or safety. Extracting, partitioning and customizing are performed to create all the necessary files for running OSM-CCTV router.

C. Privacy-mode and safety-mode

Our CCTV-aware routing solution has two alternative modes called privacy-mode and safety-mode. Their functioning utilizes the entrance and exit nodes on the ways that intersect with the fields of vision of CCTV cameras. Not allowing to traverse through these nodes and through the field of vision of a CCTV camera would maintain privacy, hence, privacy-mode is being used. Privacy-mode is implemented as a Lua profile [43] by blacklisting access tags that indicate surveillance. Safety-mode works vice versa. Safety-mode does not blacklist the access tags in the Lua profile, but it is instead enabled by using traffic updates [53] of the standard OSM functionality. These traffic updates are used to weigh the calculations so that the route will run through areas with more CCTV cameras. These calculations use weights that are added to the OSM file in the data manipulation phase before running the router.

D. Running Open Street Routing Machine

In this section, running OSM-CCTV is described. OSM-CCTV (as inherited from baseline OSM project) has a backend [54] and a frontend [55]. We have chosen to use the default OSM backend (i.e., controlling the routing via blacklist tags, weights and traffic updates), but decided fork from the standard OSM frontend [56] as it is more suitable for the needs for toggling between the privacy-mode and safety-mode. While not necessary functionality for the actual routing purposes, displaying the CCTV cameras on the map that is rendered by the OSM-CCTV frontend requires hosting its own tile server. For the tile server, we decided to use openstreetmap-tile-server by Overvoorde [57] with style modifications made by Townsend [58]. We run the services in virtual containers using Docker and Docker Compose [49]. Internal routing between Docker containers is performed with Nginx [51].

Running OSM-CCTV and handling requests works as follows. After the data processing phase, two instances of the OSM-CCTV backend and the OSM-CCTV frontend are run
within *Docker* containers. One of the two backend containers is dedicated to *privacy-mode* and the other one to *safety-mode*. Two distinct containers are necessary because the standard *OSRM* requires a static, pre-processed set of files for each different mode. A user can access the frontend with a web browser and input a typical query for requesting a route from point A to point B, also choosing either *privacy-mode* or *safety-mode* from a drop-down menu. The request is sent to a reverse proxy, *Nginx* [51], which uses the Uniform Resource Locator (URL) to forward the request to the correct backend container, depending on the condition of either *privacy-mode* or *safety-mode* being used. The backend computes the route and sends it back to be drawn on a map by the frontend that is visible on the web browser.

**IV. EVALUATION**

In this section, we present the function of our *CCTV-aware* routing solution on a synthetic map that was created for the evaluation. First, we provide screenshots from *Java OpenStreetMap Editor (JOSM)* [59] to illustrate what *OSM* files without and with cameras added by our algorithm look like. On the synthetic evaluation map, we first describe how the standard *OSRM* works by default and without either *privacy-mode* or *safety-mode*. We then proceed into showing examples of *privacy-mode* where the calculated route avoids CCTV cameras completely. Following *privacy-mode*, we demonstrate *safety-mode* in which the route is weighed to prefer surveilled routes. *Safety-mode* is not as absolute as *privacy-mode* as it does not choose too long routes even if they are surveilled by CCTV cameras. After demonstrating *privacy-mode* and *safety-mode* with a comprehensive synthetic setup (Section IV), we finally show routing examples with real-world data (Section V).

**A. Java OpenStreetMap Editor**

In this section, we provide screenshots of *OSM* data viewed in *JOSM*. The figures illustrate what *JOSM* in general and with CCTV cameras added with our script look like. The directions and the radii of the cameras are random as our data does not yet contain that data. For clarity, the borders of the areas surveilled by cameras are marked by *OSM* node objects. When actually using the router, these nodes should be omitted from the data, since they do not affect the routing.

Figure 1 and Figure 2 show an area from the center of Jyväskylä city, Finland. In Figure 1 a normal *OSM* file is examined. In Figure 2 we have used our script to add CCTV cameras to locations based on data we have gathered during our earlier feasibility-study work [39]. Fields of vision of cameras tagged as *round* can be seen as yellow circles that consist of *OSM* nodes. Here, the camera itself is illustrated with a rather randomly chosen figure, a fountain, in the center of the circle. Furthermore, yellow arches in varying sizes represent the fields of vision of cameras tagged as *directed*. For evaluation purposes, in both of *round* and *directed* cameras, the radii, the angles of view and the directions have been given random but realistic values, as our current data does not always include the real values for them. On the other hand, such a random value assignment script is also handy to quickly generate synthetic examples, and spot any bugs or inconsistencies in our routing engine and algorithms.

Figure 3 is an image of the synthetic evaluation map used for the case examples. This image illustrates the default state of the evaluation map before any data manipulation has been executed. In Figure 4 there is a *directed* CCTV camera added by a script. The ways near the camera are split in three distinct ways, all of which can be routed through. Figure 5 illustrates a similar case with more CCTV cameras added to the data.

Next, before proceeding to either *privacy-mode* or *safety-mode*, Figure 6 demonstrates the default behavior of the
standard OSRM in its frontend. In this case, the shortest possible route is being used.

B. Privacy-mode

In this section, to demonstrate how the privacy-mode of our CCTV-aware solution operates, our evaluations are presented as examples of some relevant cases. Privacy-mode aims to maintain privacy by avoiding surveilled areas. It operates in a strict way as it never routes through a field of vision of a CCTV camera. If only a part of a way is under surveillance, the route goes from the other side of that way. Here, the other side means one of the split ways parallel to the original one in the middle.

First, in Figure 7 there is an example of all the shortest ways being blocked by a CCTV camera. The calculated route will be the shortest unsurveilled one that OSRM-CCTV can find. In Figure 8, even the route used in the previous image is closed (see Figure 7) and the longest way on the map is used. Figure 9 illustrates a more complicated route where many CCTV cameras surveil different parts of the map but there is still an unsurveilled route that is shorter than the longest possible route.

In Figure 10, there is a case where every possible route is blocked by surveillance – no route is being generated and privacy is preserved. If only a part of a way is under surveillance (see Figure 11), privacy-mode allows routing through the other side of the road. This is the result of using split ways to represent the dimensions of ways. Figure 12 shows a zoomed-in portion of area partly under surveillance where OSRM-CCTV routes through the split way on the northern side of the road (see Figure 11).

C. Safety-mode

In this section, the safety-mode of our CCTV-aware solution is evaluated. The results are presented with similar case examples than earlier in the case of privacy-mode. A safe route in safety-mode means a route that is under the surveillance of CCTV cameras. In principle, a route leading through the fields of vision of more cameras would mean a safer route. Safety-mode is not simply the privacy-mode reversed and it does not always choose a surveilled route, but it works in a more selective manner. The standard OSRM [40], [60] provides traffic updates [53] which are used to weigh preferable routes based on the amount of CCTV cameras surveilling the way. If a route would be too long, the weighing can cause OSRM-CCTV to generate a route that has no camera surveillance.

Figure 13 demonstrates a simple example where one of the ways is under the surveillance of three CCTV cameras.
OSRM-CCTV) chooses that way based on the weighing performed according to the OSRM-CCTV traffic updates [53]. In Figure 14 a similar case is displayed to prove that the chosen route is the surveilled one. In Figure 15 one of the ways is surveilled by one more cameras than the next one. OSRM-CCTV chooses the route that has more CCTV cameras surveilling it. Following Figure 15 Figure 16 shows a similar case to prove that the chosen route is the one under the surveillance of more CCTV cameras. Figure 17 demonstrates the same case seen earlier with privacy-mode where only part of a way is under surveillance (see Figure 11 in last section, which is clarified with Figure 12). In this case, the difference is that instead of using privacy-mode, safety-mode is being used. The small nudge after the diagonal part of the way indicates OSRM-CCTV routing to a split route which leads through a longer surveilled part of the way than the other options.

Lastly, as shown earlier, Figure 15 and Figure 16 demonstrate that traffic updates allow OSRM-CCTV to prefer a surveilled route over a short one, and that OSRM-CCTV chooses a route that is under the surveillance of more CCTV cameras. Figure 18 shows OSRM-CCTV choosing the shortest route instead of following too long a route. To some extent, this is intended behavior since one would not likely prefer to use an arbitrarily long route only to reach as many CCTV cameras as possible. However, in our evaluations we could not determine exactly how OSRM-CCTV with its traffic updates calculates the preference between the route length and the weights of the route. For example, choosing an arbitrarily large rate value (see standard OSRM Traffic [53]) one would expect OSRM-CCTV to route through a correspondingly long route as long as it has many CCTV cameras surveilling it. As that does not seem to be the case, fine-tuning the safety-mode would require more evaluation.

V. REAL-WORLD EXAMPLES

In this section, we present five examples of how privacy-mode and safety-mode of our CCTV-aware approach operate in the downtown of the city of Jyväskylä, Finland. This location was chosen because we have mapped around 450 CCTV cameras around the city of Jyväskylä [39]. The mapped data contains for example the coordinates and the type of the cameras. The type can be either round or directed. The round type corresponds to a camera with a circular, 360 degree angle of view. The directed type corresponds to a camera with a less than 360 degree angle of view. In our current data and for the purposes of our evaluations, the radii, the angles of view and the directions have been added randomly with a script.
Three of the five examples are depicted as a pair of images, both of which have the same starting and ending point. The starting point is marked with a green marker and the endpoint is marked with a red marker. Regarding the first three examples, the difference between the images in each pair is that the first one operates on privacy-mode and the second one on safety-mode. The last two of the five examples compare the default mode of OSRM-CCTV routing and the safety-mode of our CCTV-aware approach. These two are compared to demonstrate that in safety-mode, the router will prefer the longer route over the default route that is shorter, as long as the calculated route is not too long.

In the first example, Figure 19 shows a route through the middle of the downtown of the city of Jyväskylä. Privacy-mode is being used, and OSRM-CCTV does not route through the field of vision of any CCTV cameras. Figure 20 shows the same scenario run with safety-mode. The route goes straight through the fields of vision of multiple CCTV cameras.

In the second example, privacy-mode is run in Figure 21. Starting from the green marker, OSRM-CCTV first avoids the CCTV cameras right at the start of the route by switching to the other side of the route. After that the route leads to the endpoint marked with a red marker, avoiding all possible CCTV cameras. In Figure 22, OSRM-CCTV operates on safety-mode and the route goes through the middle of the map and many CCTV cameras.

The third example shows a situation that is different than the earlier two. Figure 23 demonstrates how running OSRM-CCTV in privacy-mode avoids all possible cameras until the end of the route. Since the endpoint is in between of multiple CCTV cameras and it is impossible to route there without travelling through the field of vision of a CCTV camera,

OSRM-CCTV routes as close as it can but no further. Privacy is not compromised as the route does not go through the field of vision of any CCTV cameras. For comparison, running safety-mode in Figure 24 the route goes from the starting point straight to the endpoint, making some turns to reach as many fields of vision of CCTV cameras as possible.

Next, the last two examples are demonstrated between the default mode of OSRM-CCTV routing and the safety-mode of our CCTV-aware approach. These are compared to show that in safety-mode the router will prefer the longer route over the default, shorter route. In Figure 25 when the default mode of OSRM-CCTV routing is used, there is a situation where OSRM-CCTV chooses the shortest, 277-meter route by default.
Fig. 22: OSRM-CCTV run with safety-mode chooses the its route to route through the fields of vision of multiple CCTV cameras in the downtown of Jyväskylä, Finland.

Fig. 23: OSRM-CCTV run with privacy-mode avoids all possible CCTV cameras but does not route to the endpoint on the left side of the image. Avoiding fields of vision of CCTV cameras would be impossible there, hence privacy is maintained and the endpoint is reached only as closely as possible.

instead of the longer, 351-meter alternative route seen in shaded purple color. When safety-mode is on (see Figure 26), conversely, OSRM-CCTV chooses the longer route that leads through the areas of vision of multiple CCTV cameras.

Finally, the last example is similar to the previous one. When running on safety-mode, OSRM-CCTV chooses the longer, 366-meter route (see Figure 28) instead of the shorter 320-meter route that is chosen by default (see Figure 27). The alternative route seen in shaded purple color is different than in the case without CCTV cameras (see Figure 27) because also the alternative route leads through the fields of vision of CCTV cameras. The chosen route is weighed to be more preferable than the alternative one.

Fig. 25: Running on the default routing mode, OSRM chooses the shortest route.
VI. CONCLUSION

In this paper, we propose, implement and evaluate a CCTV-aware privacy- and safety-oriented routing solution. To the best of our knowledge, this is the first work where a CCTV-aware approach to routing has been conceptualized and proposed. For the routing engine, we start from existing open-source Open Street Routing Machine (OSRM) routing framework. Subsequently we customize it, and open-source release OSRM-CCTV – a CCTV-aware routing modification based on OSRM. We also validate and demonstrate the effectiveness and usability of the system on a handful of synthetic and real-world examples. To help validate our work and to further encourage the development and wide adoption of the system, we release the relevant artifacts (e.g., code, data, documentation) as open-source: https://github.com/Fuziih

ACKNOWLEDGMENTS

We acknowledge grants of computer capacity from the Finnish Grid and Cloud Infrastructure (FGCI) (persistent identifier urn:nbn:fi:research-infras-2016072533). Part of this research was kindly supported by a “17.06.2020 Decision of the Research Dean on research funding within the faculty” grant from the Faculty of Information Technology of the University of Jyväskylä, and the grant was facilitated and managed by Dr. Andrei Costin. The authors would also like to thank the following persons for their dedicated efforts during the crowdsourced data collection and/or internal paper review phase(s): Syed Khandker (special mention and many thanks).

Hannu Turtiainen would like to also thank the Finnish Cultural Foundation / Suomen Kulttuurirahasto (https://skr.fi/en) for supporting his PhD dissertation work and research (grant decision 00211119), and the Faculty of Information Technology of University of Jyväskylä (JYU), in particular Prof. Timo Hämäläinen, for partly supporting his PhD supervision at JYU during 2021–2022.

REFERENCES

[1] P. Bischoff, “Surveillance Camera Statistics: Which City has the Most CCTV Cameras?” https://www.comparitech.com/blog/vpn-privacy/the-worlds-most-surveilled-cities/ May 2021.
[2] E. Cosgrove, “One billion surveillance cameras will be watching around the world in 2021.” https://cnbc.com/2019/12/06/one-billion-surveillance-cameras-will-be-watching-globally-in-2021.html
33 “How Many CCTV Cameras in London?” https://www.caughtoncamera.net/news/how-many-cctv-cameras-in-london/

34 B. Karas, “Americans Vastly Underestimate Being Recorded on CCTV.” https://ipvm.com/reports/americas-cctv-recording/

35 J. Pasley, “I documented every surveillance camera on my way to work in New York City, and it revealed a dystopian surveillance reality.” https://www.businessinsider.com/how-many-security-cameras-in-new-york-city-2019-12?r=US&IR=T

36 A. von Hirsch, The ethics of public television surveillance. Ethical and Social Perspectives on Situational Crime Prevention. Hart Publishing, 2000.

37 W. Hu, T. Tan, L. Wang, and S. Maybank, “A survey on visual surveillance of object motion and behaviors,” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 34, no. 3, pp. 334–352, 2004.

38 F. W. Wheeler, R. L. Weiss, and P. H. Tu, “Face recognition at a distance system for surveillance applications,” in 2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS). IEEE, 2010, pp. 1–8.

39 Axis, “Identification and recognition,” https://www.axis.com/files/feature_articles/ar_id_and_recognition_53836_en_1309_lo.pdf.

40 B. v. S.-T. Larsen, Setting the watch: privacy and the ethics of CCTV surveillance. Bloomsbury Publishing, 2011.

41 A. Costin, “Security of CCTV and video surveillance systems: Threats, vulnerabilities, attacks, and mitigations,” in 6th International Workshop on Trustworthy Embedded Devices (TrustED), 2016.

42 OpenStreetMap Wiki, “Routing online routers,” https://wiki.openstreetmap.org/wiki/Routing/online_routers

43 ———, “Routing offline routers,” https://wiki.openstreetmap.org/wiki/Routing/offline_routers

44 OpenStreetMap Wiki, “List of OSM-based services — OpenStreetMap Wiki,” https://wiki.openstreetmap.org/wiki/OSM-based_services&oldid=2052956, 2020, [Online; accessed 1- November–2020].

45 D. Luxen and C. Vetter, “Real-time routing with OpenStreetMap data,” in Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems, 2011.

46 H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck, “Route planning in transportation networks,” in Algorithm engineering. Springer, 2016.

47 R. J. Szczepańczuk, P. Galkowski, I. S. Glicktein, and N. Ternullo, “Robust algorithm for real-time route planning,” IEEE Transactions on Aerospace and Electronic Systems, vol. 36, no. 3, pp. 869–878, 2000.

48 H. Turița, A. Costin, T. Hamalainen, and T. Lahtinen, “Towards large-scale, automated, accurate detection of CCTV camera objects using computer vision. Applications and implications for privacy, safety, and cybersecurity.(Preprint),” arXiv preprint arXiv:2006.03870, 2020.

49 H. I. F. T. H. GIScience, https://openrouteservice.org/ [2020].

50 P. Kasemupphorn and H. A. Karimi, “Personalised routing for wheelchair navigation,” Journal of Location Based Services, vol. 3, no. 1, pp. 24–54, 2009. [Online]. Available: https://doi.org/10.1080/17489720923873926

51 A. Zipf, A. Mobasher, A. Rousell, and S. Hahmann, “Crowdsourcing for individual needs—the case of routing and navigation for mobility-impaired persons,” European Handbook of Crowdsourced Geographic Information, pp. 323–337, 2016.

52 A. Franco, “What is curvature?” https://roadcurvature.com/ [2016]. [Online; accessed 1- November–2020].

53 C. Olaverri Monreal, M. Pichler, G. Krizek, and S. Naumann, “Shadow vulnerabilities, attacks, and mitigations,” in Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ser. GIS ’11. New York, NY, USA: ACM, 2011, pp. 513–516. [Online]. Available: http://doi.acm.org/10.1145/2093973.2094062.

54 D. Luxen, “[OSM-dev] Announcing the immediate availability of the Open Source Routing Machine,” https://lists.openstreetmap.org/pipermail/dev/2010-July/019834.html, 2010.

55 R. J. d'souza, W. C. Lesley, and L. H. de Figueiredo, “About,” https://www.lua.org/license.html.

56 Project-OSRM, “Osm profiles,” https://github.com/Project-OSRM/osrm-backend/blob/master/docs/profiles.md, 2020.

57 G. van Rossum, https://www.python.org/.

58 J. Topf, “Osmium Library,” https://osmcode.org/libosmium/

59 S. Hoffmann, “Osmium python library,” https://osmcode.org/pyosmium/

60 S. Gillies, “The shapely user manual,” https://shapely.readthedocs.io/en/stable/manual.html.

61 J. Whitaker, https://pypi.org/project/pyproj/.

62 C. Olaverri Monreal, M. Pichler, G. Krizek, and S. Naumann, “Shadow vulnerabilities, attacks, and mitigations,” in Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ser. GIS ’11. New York, NY, USA: ACM, 2011, pp. 513–516. [Online]. Available: http://doi.acm.org/10.1145/2093973.2094062.

63 R. J. d’souza, W. C. Lesley, and L. H. de Figueiredo, “About,” https://www.lua.org/license.html.

64 Project-OSRM, “Osm profiles,” https://github.com/Project-OSRM/osrm-backend/blob/master/docs/profiles.md, 2020.

65 G. van Rossum, https://www.python.org/.

66 J. Topf, “Osmium Library,” https://osmcode.org/libosmium/

67 S. Hoffmann, “Osmium python library,” https://osmcode.org/pyosmium/

68 S. Gillies, “The shapely user manual,” https://shapely.readthedocs.io/en/stable/manual.html.

69 J. Whitaker, https://pypi.org/project/pyproj/.

70 J. Whitaker, https://pypi.org/project/pyproj/.

71 Baidu, map.baidu.com.

72 Yandex, https://yandex.eu/maps/.

73 Highways England, “Traffic,” https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/456888/traffic-flow-data.pdf.

74 OpenStreetMap Wiki, “Way — openstreetmap wiki,” https://wiki.openstreetmap.org/wiki/index.php?title=Way&oldid=2045405, 2020. [Online; accessed 2-November–2020].