GEO-TAGGED IMAGE RETRIEVAL FROM MAPILLARY STREET IMAGES FOR A TARGET BUILDING

Naime Çelik [1], Emre Sümer [1]
[1] Dept. of Computer Engineering, Baskent University, Ankara, Turkey - 21610281@mail.baskent.edu.tr, esumer@baskent.edu.tr

Commission VI, WG VI/4

KEY WORDS: Image Retrieval; Target Building; pre-trained Unet; Mapillary; street view images

ABSTRACT:

This study aims to investigate the possibility to automate the image selection process for the target building from Mapillary images through a web application where the user only initiates one image of the target building as a query. Using the data provided with Mapillary API and Overpass API, all images having full or partial coverage of the target building were selected. Then the images were segmented by using a pre-trained U-Net model to discard any images having less than 20% building coverage. The experiments showed promising results yielding 0.971 and 0.887 of overall accuracy after segmentation steps for two different target buildings.

1. INTRODUCTION

Recently Mapillary and OpenstreetCam have been providing crowdsourced street-level images via volunteers. One can follow sequences of connected georeferenced images and navigate through images via these services. In the background, both services are running computer vision algorithms as an incentive for extracting information as commercial products (Leon & Quinn, 2019). Mapillary allows users to download their crowdsourced street-level images through their API (Mapillary, 2019). All of the EXIF information is lost upon downloading images, although this API provides partial information of the EXIF in JSON format to the users.

In this study, we have evaluated Mapillary's image properties and experimented to automate the image selection process of the target building. A flask web application was developed to retrieve public images from Mapillary API and also utilizing Overpass API (2020) containing Open Street Map (OSM) data. As a first goal, images having full and partial coverage of the target building are selected, initially based on their location and camera angle. Secondly, segmentation from the pre-trained U-Net model is used to select images of the target building that have sufficient coverage of the building within the image.

By providing one query image of the target, one can automatically retrieve almost all images of the target with full or partial coverage with the information of the relative location of the target in the image (right, left, or direct). Images occluded by other buildings can be detected and eliminated as well. An exemplary use of a pre-trained U-Net model in this study demonstrates promising results to further narrowing down image selection to retrieve images with higher pixel content of the target building.

As the use of crowdsourced street-level image platforms and the number of images contributed by the volunteers to these platform increases, and foreseen to be increased, the community should be able to find a way of data retrieval for a specific target. This study provides a method that can be implemented by other researchers with publicly available data from Mapillary.

2. RELATED STUDIES

2.1 Image Selection for a Target

Araujo et al. (2015) used a query image of a building to identify panoramic views of the interest from Google street view panoramic sequences. The graph sequence of images was searched with Breadth-First Search (BFS) strategy. Their approach uses Affine-SIFT features to detect target building in the panoramic images. 15 cases out of 30 were failed due to issues such as incorrect feature matching, GPS coordinate errors, and cases with images taken at night and obscured facades with trees. Another study (Wolff et al., 2016) was detecting street images of target building facade in oblique aerial images by matching context information such as color, shape, and spatial similarity to those in the street images. Cheng et al. (2018), implemented target image selection in their study and used a structured organized image dataset from Tencent Street Views Images as a reference to geotag images from unidentified resources. The crawled images were processed to exact SIFT features and to establish an index. Each query image was compared with their extracted features. The images with the best match were used to generate 3D reconstruction with Structure from Motion (SfM) as all images in the reference database taken with one fixed camera. Their goal was to acquire accurate query image location and the query image was geo-retagged with 69m accuracy by using parameters acquired from 3D reconstruction results.

Krylov & Dahyot (2018) used Mapillary street images for geolocating target objects (traffic lights) and the acquired images within a specified field of view were enhanced with estimated image bearings and camera positions by using OpenSFM. Their pipeline includes semantic segmentation and monocular depth estimation with Convolutional Neural Networks (CNN) on street view images to detect target objects. Any images with no object or no paired images were discarded and the target object location was determined with triangulation.

When images are uploaded by a user into Mapillary app, Mapillary runs SfM algorithm onto images and computes new latitudes and longitudes, aligns images based on outputs, Mapillary JavaScript library provides functions to get computed new positions and much accurate bearing angles (Lorenzon, 2019). The computed new bearing angles are more accurate than those in the provided EXIF information and in this study, the computed locations and computed camera angles are requested from Mapillary API. Although, the calculated position and directions by Mapillary also have errors as they are indicated in

the discussion section and the image retrieval errors caused by erroneous directions need to be mitigated as a future study.

2.2 Eliminating Occluded Images

Another issue is to eliminate images with possible occlusions such as trees, other buildings, or cars or with low visibility of the target. To find ideal images, further image processing, and testing are necessary on the usability of the image.

CNN based image segmentation has been widely studied to understand scene content with methods differ in feature extraction as VGG16, Resnet18, MobileNet, and ShuffleNet and in decoding as, U-Net and Dilation (Siam et al., 2018).

U-Net model may provide 85% accuracy on buildings in the Cityscapes dataset. There are also ResNet based CNN implemented by Mapillary which provides 90% accuracy on buildings of the same dataset (Bulò et al., 2017), also Porzi et. al (2019) provides PyTorch implemented ResNet-50 with a Feature Pyramid Network (FPN) model and made it available on GitHub. Image segmentation with CNNs has been already presented by Mapillary and object detection within an image is available with a commercial license, allowing the user to request images based on their pixel label. Similarly, the percentage of building pixel coverage was taken into consideration to discard images in this study.

3. MATERIAL & METHODS

3.1 Data

Crowdsourcing images from Mapillary has been used in this study. The data on this platform was collected through cell phones and other cameras with GPS.

https://a.mapillary.com/v3/images?client_id=<client_id>
&closeto=<Lon>,<.Lat>&computed_coordinates=&includes=cca (a)


```
"type": "FeatureCollection",
"features": [{
"type": "Feature",
"properties": {
  "original_coordinates": [30.4975131, 39.7642607],
  "ca": 146.0753173828125,
  "camera_make": "Apple",
  "camera_model": "iPhone 11",
  "captured_at": "2020-01-03T13:42:47.316Z",
  "key": "Pn63FBYqz7w6Jb5DSw04Ng",
  "pano": false,
  "sequence_key": "nzsfsx7yd7r2wg5mtm7yqo",
  "user_key": "zYhfeYWOZKm4dOzEKcG-Nw",
  "username": "ncelik",
  "cca": 342.27526993997697},
"geometry": {
  "type": "Point",
  "coordinates": [30.4975053, 39.7642728]] }},
```

(b)

Figure 1. A query URL for Mapillary API (a) and downloaded JSON format response from Mapillary of one of the image (b)

Mapillary API (2019) allows a user to access and download image data through HTTP endpoints. The API also provides partial EXIF information in JSON format. (Detail regarding EXIF format can be found at https://exiftool.org/TagNames/EXIF.html) Each image located in https://images.mapillary.com/{key}/thumb-2048.jpg URL and available with CC BY-SA license allowing individual and educational use.

```
[out:json];
way(39.76354600, 30.49646200, 39.76498900,30.49833900)
[building]; (._;>;);
out;
```

Figure 2. Sample query for Overpass API returns building data within specified coordinates (the extent of the map in the web application)

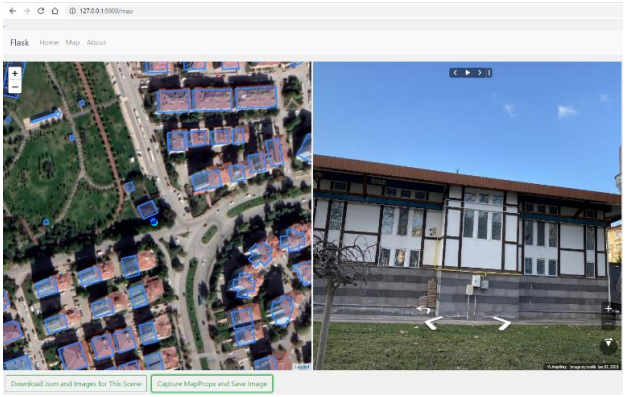

Figure 3. The basic user interface of the flask web application

Overpass API was also used for acquiring vector format OSM buildings to identify possible target buildings. The query landmark was specified as "building" (Figure 2) and the area of query was restricted with the map corner coordinates of the user's map view of the developed flask web application. Figure 3 shows a sample retrieved buildings from Overpass API.

3.2 Method

The developed Flask web application running in the Docker-based architecture consumes Overpass API and Mapillary API allowing the user to select a street view image on Mapillary-JS (MapillaryJS, 2020) image sequence. Based on the selected image location, all images with the Mapillary default limit (200 images) are downloaded, renamed with their key name inside the folder that named with its sequence_key. The EXIF information of images are also modified with the information provided in JSON data from Mapillary API during this process. Computed camera angle (cca) is included, and the location of computed longitude and latitude values are used to modify image EXIF location parameters (GPSLongitude, GPSLatitude).

PostGIS database in a docker container is used to store overpass vector data containing buildings within the map area. The user selected image location and downloaded image locations are also stored in the same database. Once the images are processed by the application, the selected images are returned to the user with HTTP endpoints.

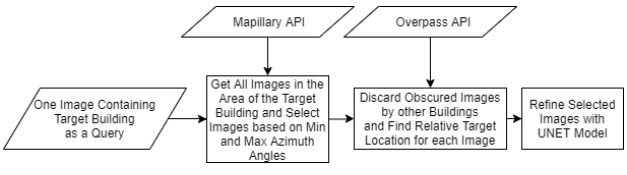

Figure 4. General workflow for the web application

3.2.1 Identifying the Target Building: The process for selecting images of the target building starts with a query image selected by the user. The closest building along the ± 90 cca direction is selected as the target building in the query image.

3.2.2 Calculating Minimum and Maximum Azimuth for Each Image: The azimuth angle is calculated between simplified target exterior points and image location. The points providing the maximum and minimum azimuth angles are selected. If the image cca value is within these angles, the image is accepted for further processing. Cheng et al. (2018) also used similar azimuth angle methodology in their study for image selection. Original locations and camera angles are much more prone to errors due to incorrect GPS or compass measurements in the volunteer's devices. Therefore, requested cca and computed image locations are used for each image from Mapillary API.

Figure 5 shows the process flow for finding the direct case images where cca values within the target minimum and maximum azimuth angles. For each image, the calculated minimum and maximum azimuth of the target exterior points are evaluated and checked for their coverage of image cca value provided by Mapillary API. Figure 6 also illustrates a sample image with its cca value that is not between the minimum and maximum azimuth of the target and the map showing the location of the image, relative to the buildings.

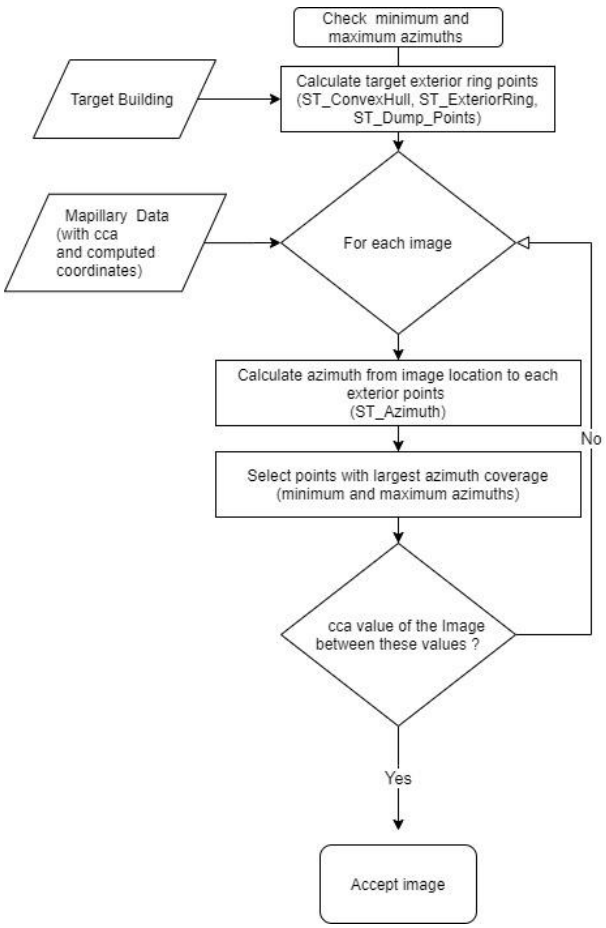

Figure 5. Process flow for checking the minimum and maximum azimuth of an image

As the distance from the target to the image location increases, the angle difference between the minimum and maximum azimuth decreases. If the target is not directly located along the image direction, the image will be discarded. Figure 6 shows an example of this situation. An angle tolerance is necessary to avoid this problem and detect images that may contain the target on the right or left part of the image.

Figure 6. The map of image location relative to the target building

3.2.3 Utilization of Angle Tolerance: Images may contain the target building that is not directly along the cca direction as the sample image in Figure 6. To detect any images as not a direct case, the cca direction should be flexed with an angle tolerance. The line projected from image location with cca value can be used to calculate two angles (β1 and β2) in Figure 7. Most cameras with a focal length of 24 mm to 70 mm have less than 90-100° view angle (Panasonic, 2020). If the target is located in the left or the right side of the image, by restricting (β1 and β2) with an angle such as 45° or 50°, the tolerance angle necessary for each image can be calculated.

In the additional process flow for calculating angle tolerance (Figure 8), the relative location of the building can also be calculated with one side buffer by using a buffer area with the largest distance to the exterior points. If all exterior points are placed within the left buffer area, the target will be located on the left side of the image, else on the right.

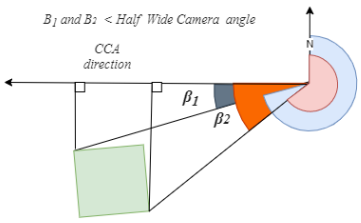

Figure 7. Tolerance angle necessary for each image can be found restricting by β1 and β2

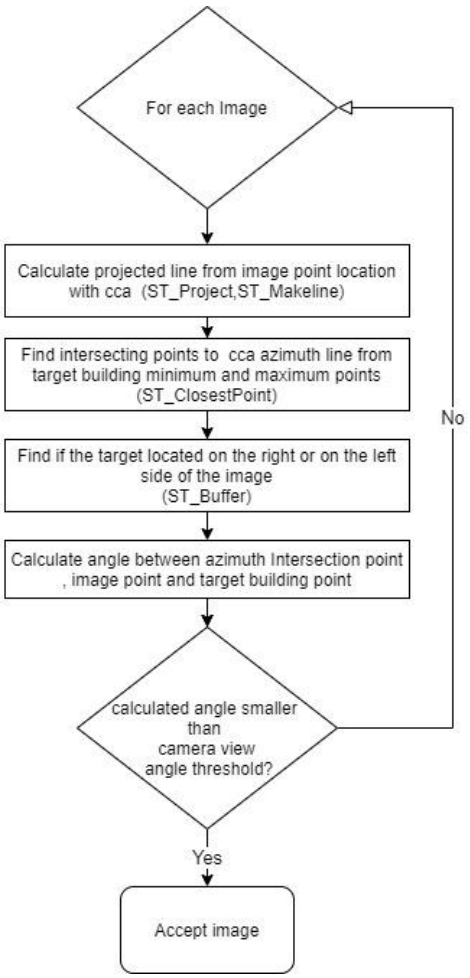

Figure 8. Additional process flow for calculating necessary tolerance angle for the images

3.2.4 Discarding Images with Buildings that Occludes Target Building: To check if any other building is obscuring the image from viewing the target building, the exterior points of the target building are used. If both lines from points providing the minimum and maximum azimuth of the target intersect with any building other than the target itself, then that image is rejected as an obscured image. Figure 9 shows a case of an occluded image.

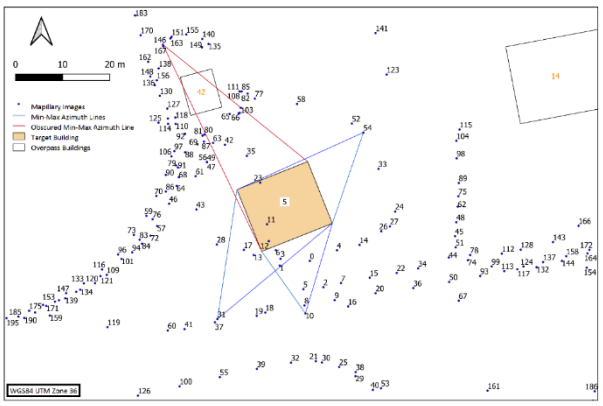

Figure 9. The image 146 obscured by building id: 42, as the minimum and maximum azimuth lines of the target building id: 5 intersect this building, Images of 37, 54 and 10 are not intersecting. (Illustration with only 4 images, the process flow takes all images into consideration)

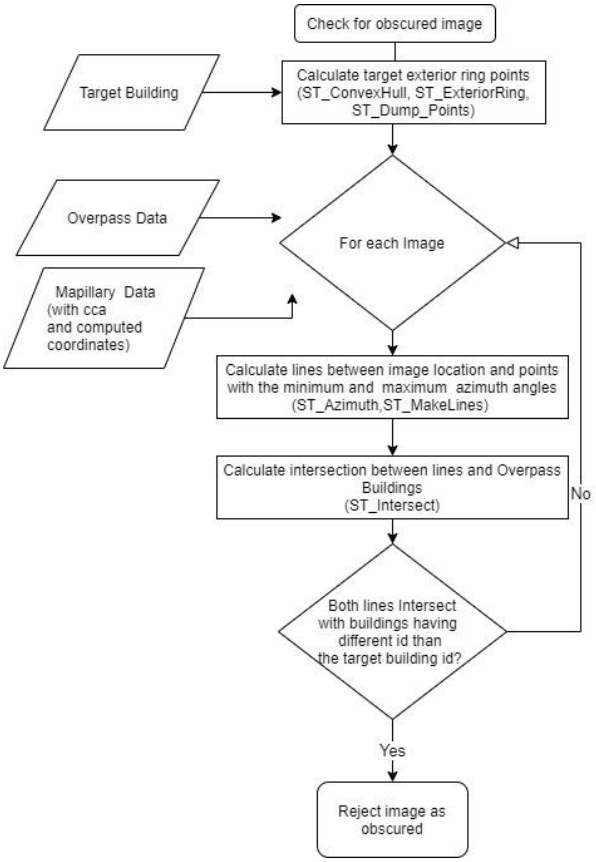

Figure 10. Process flow for checking if the images obscured by another building

3.2.5 Discard Images with Insufficient building coverage using Unet CNN: As the target may be concealed by an object such as trees, cars, or people, to eliminate images that have insufficient coverage of the building, pre-trained Keras U-Net (Humbarwadi, 2020) model previously trained with Mapillary Vistas Dataset is used. The model was trained for 150k iterations with a learning rate of 0.004 with 256x256 images with 64 filter and 66 classes. The model encodes an image with 3x3 convolutional kernel with two 2d convolution operation in each convolution block and apply 2x2 max pooling operation to reduce to the size of the feature map thus, downsamples the image. Transposed convolution in deconvolution block upsamples image with the corresponding convolutional layer to acquire an increased resolution on the final output. Although PyTorch implementation was provided by Mapillary (Porzi et al., 2019), due to the usage of CPU based platform with no CUDA in our experiment, TensorFlow Keras implementation was preferred in the current study. The repository of the pre-trained model was missing the necessary config.json file containing 66 object segmentation labeling and it was obtained via Mapillary Vistas Research Dataset (Neuhold et al., 2017) requested through Mapillary web page.

Pixels labeled as building are masked and the pixel ratio of the connected components in the masked image is calculated as coverage ratio. Figure 11 shows an image of a direct case, where the calculation of the building coverage ratio takes all connected pixels into account and results in 0.49. Figure 12 shows an image where the relative target location (right, left, or direct) identified as right. The calculation takes into the only right side of the image and the building coverage ratio results as 0.0 since the target is behind a tree.

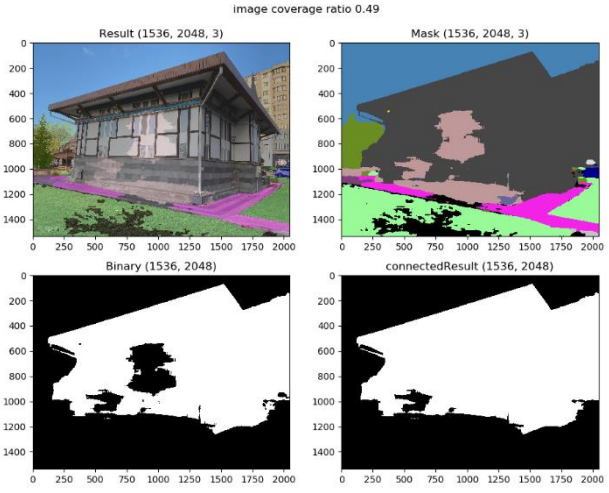

Figure 11. An image (image key: 32pUYPVcaPCp0TMWdg_SVA) with building coverage ratio 0.49

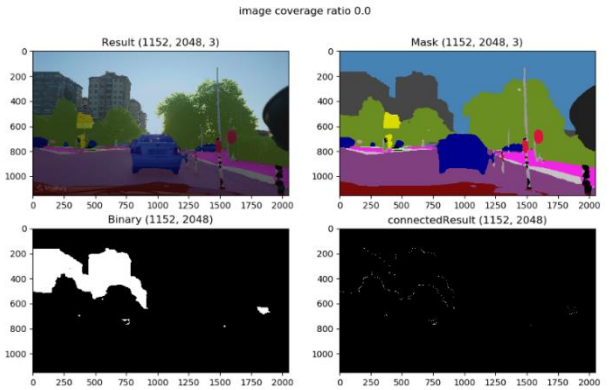

Figure 12. An image (image key: 2FZgFFC3jVJnxolqrdUv-Q), the relative target location was detected as right, provides only pixels on the right side of the image on coverage ratio calculation, thus results coverage ratio 0.0

3.3 Reference Data

Two target buildings were selected in a public area (Dede Korkut Park, Eskisehir), shown in Figure 9 with building ids 5 and 42. Various images of the targets were collected and uploaded through Mapillary App with two different mobile phones. The images of the targets were requested with the default image request size of 200 through developed flask web application. The downloaded images were uploaded by the authors and also other unknown volunteers. The images were evaluated and requested on May 24, 2020. The images uploaded or modified later of this date in Mapillary were not considered in the evaluation.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3)$$

$$Accuracy = \frac{TP + TN}{\mathrm{TP + TN + FP + FN}} \quad (4)$$

Each image was evaluated manually on their content and marked as True Positive (TP) if image contains the target fully or partially within image view and accepted, False Positive (FP) if image does not contains the target within the image view but accepted, True Negative (TN) if image does not contain the target and rejected, False Negative (FN) if image contains the target fully or partially but rejected. Additionally, the metrics; Precision, Recall, F-score and Accuracy were also computed given in the following equations (1) to (4).

4. RESULT AND DISCUSSION

	TB:5 Th:None	TB:5 Th:50°	TB:5 Th:45°	TB:5 Th:40°	TB:5 Th:50° Unet
Precision	1.000	0.957	0.984	0.984	0.913
Recall	0.406	0.957	0.899	0.884	1.000
F score	0.577	0.957	0.939	0.931	0.955
Accuracy	0.777	0.967	0.957	0.951	0.971
TP	28	66	62	61	21
FP	0	3	1	1	2
TN	115	112	114	114	46
FN	41	3	7	8	0
Total	184*	184*	184*	184*	69
	TB:42 Th:None	TB:42 Th:50°	TB:42 Th 45°	TB:42 Th:40°	TB:42 Th:50° Unet
Precision	0.903	0.758	0.807	0.833	0.857
Recall	0.549	0.904	0.902	0.882	0.818
F score	0.683	0.825	0.852	0.857	0.837
Accuracy	0.857	0.890	0.912	0.918	0.887
TP	28	47	46	45	18
FP	3	15	11	9	3
TN	128	115	120	122	37
FN	23	5	5	6	4
Total	182*	182*	182*	182*	62

Table 1. Evaluation for Target Building 5 and Target Building 42 with different camera view angle thresholds and final unet segmentation with Threshold 50° (* The images with wrong cca values excluded from the calculation to evaluate model accuracy. The necessary mitigation is aimed to be a future study for images with wrong cca values.)

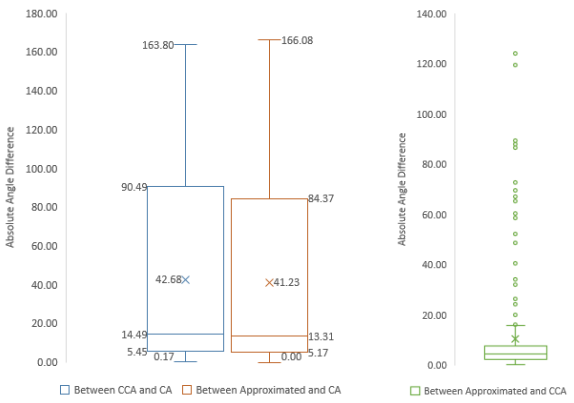

Figure 13. Box plots of absolute angle difference from cca, ca and approximated angles for 237 images obtained via Mapillary API of the two targets (some images were common for targets)

The requested cca values also had outliers. For images used in this study, approximated angles were inferred via digital protractor on web map and compared to ca and cca values. The angle difference between ca and cca shows a large discrepancy in absolute minimum angle difference. The cca values acquired from Mapillary API improves the accuracy of the camera angles, although 25 out of 237 images were incorrectly calculated on 15.92° outlier cut-off (Figure 13).

Accepted image keys with more than 30° error in their cca	cca	ca	Approx.	Diff Appx. and CCA
8RQ34MT_AxBWJdgbBNWavQ	39.189	0.000	332	67.189
JpyBYW_nwiU7a3WXJI2JGg	267.939	0.000	300	32.061
KKRHwUVTpDr1cuaSy5UnRw	300.742	0.000	335	34.258
MqYNnx16T7-aWNx9GxE83w	307.279	266.345	255	52.279
OAjnSWcOtz71yRU5IzG1xQ	211.666	99.236	300	88.334
OrPaa_JxwiiMUV-Pk6fctg	313.544	265.908	255	58.544
pCrYRJzY_iX_hr7bNmxuuA	303.589	274.291	255	48.589
SRMi20SAj9J9WZM80HLoGQ	195.861	303.043	320	124.139
5bwl3xppiWj0-rW5p8tdUA	330.649	301.955	290	40.649
emtqR3jkSBeiY68H-gR5hA	195.440	211.526	315	119.560
ovRJMaDBswbZONzHC5dqpw	327.744	281.249	255	72.744
Ozk6BRZboELAaaL7KFejfg	324.686	268.313	255	69.686
r6cnsDko560Flwt9SE79pg	315.282	268.097	255	60.282
Soz_uFvmwMGSmpMDRh5xnQ	341.551	263.469	255	86.551
yGsOdMu06-221mDQ8pQO3Q	320.488	262.600	255	65.488
YUi-3eJYplbGnsQzWKdqpg	195.100	212.327	315	119.900

Table 2. Image keys revealed with more than 30° angle difference for TB:5 and TB:42

18 images of those outliers (in Figure 13), revealed with more than 30° angle difference, were not included in the result evaluation to measure the performance of the model. Table 2 indicates the angle difference of 16 of those images affected the selection results for TB:5. The mitigation of those outliers will be discussed as a future study.

The first selection of images was based on their cca directions, whether the calculated target minimum and maximum azimuth angle values encompass the image cca directions directly. For the first target building TB:5, with no tolerance, only images directly seeing the target were retrieved with high precision but very low recall and f-score, 0.406 and 0.577 respectively. Figure 14 shows directly selected images for TB:5. Similarly, the recall and f-score for TB:42 were also low without flexing angles, which are 0.549, and 0.683 respectively.

Figure 14. Images selected for target building 5 with no angle view threshold

For TB: 42, three images were false positive and the precision is less than the precision of the other target building. These three images (blue points) of TB:42 were not fully blocked and maximum azimuth lines were not intersecting with another building (the thicker red and green lines in Figure 15). The occlusion algorithm does not discard the image if only one of the lines is intersecting. However, these images were classified as "no target" on the reference data since no part of the target building is visible due to modification with an additional tarp next to the building (enclosed by yellow rectangle).

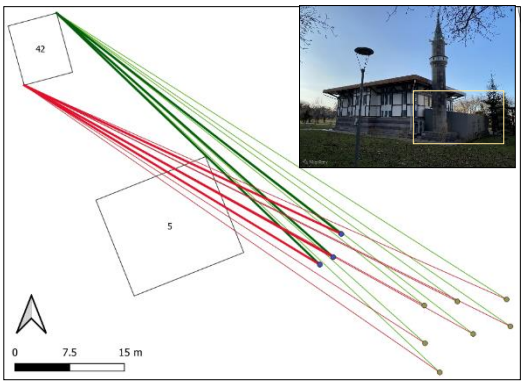

Figure 15. A misclassified reference data as "no target" (Image Id: 7VbGR_XLStmSbXDtUA4llQ) due to tarp

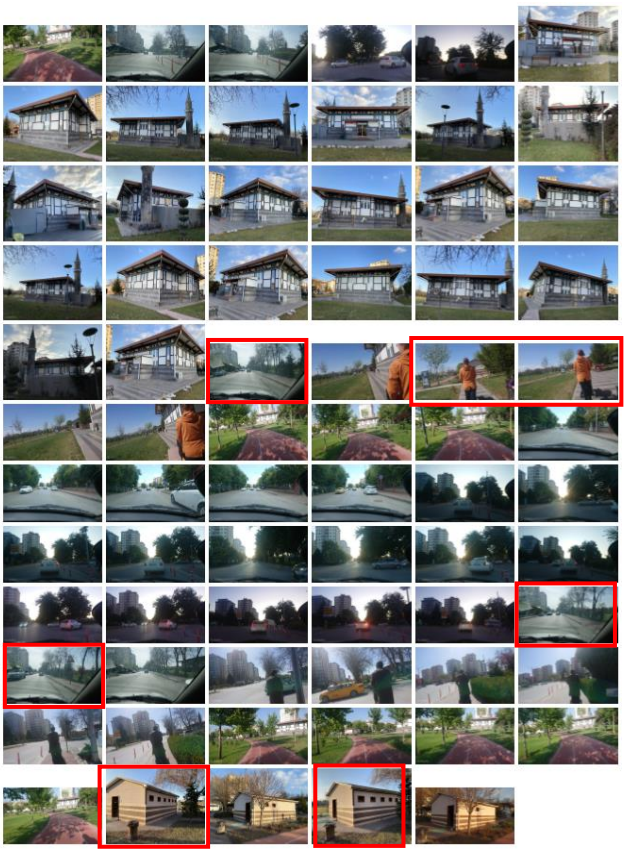

Figure 16. Images selected for target building 5 with 50 °camera view angle threshold, images enclosed with red color are those selected due to wrong cca values indicated in Table 2.

Flexing minimum and maximum angle with a camera view angle threshold increased f-score for both target buildings. After testing with different thresholds for TB:5, 50° threshold was providing f-score of 0.95, 3 images were labelled as false positive, not seeing the TB:5 and also 3 images were missed (orange borders shows the part of the target in the Figure 17).

The lower camera view angle threshold increases the false negative that is the number of missed images. Figure 16 shows all selected TB:5 Th:50, red border images are those indicated in

Table 2 negatively affecting the image selection results due to wrong cca values.

With 50° view angle threshold for TB:42, the false positive images are much higher compared to TB:5 due to tarp next to the mentioned building. Additional 7 images have the same issue mentioned above (thinner lines in Figure 15), resulted in false positive case. For the other 5 images, the target cannot be seen with this threshold and lower view angles thresholds are more suitable. As the angle threshold decreases, some of the 7 images (with tarp) were not retrieved thus lead to the lower false positive for TB: 42 with thresholds 45° and 40°.

Figure 17. Two out of three missed images for TB:5 with Th:50 °, the target can be seen very slightly (Image ids: xfUfnxDjtAYXpEmytJ8FhA and tKo5Yr0hn0zTRv5j5YGRmA respectively)

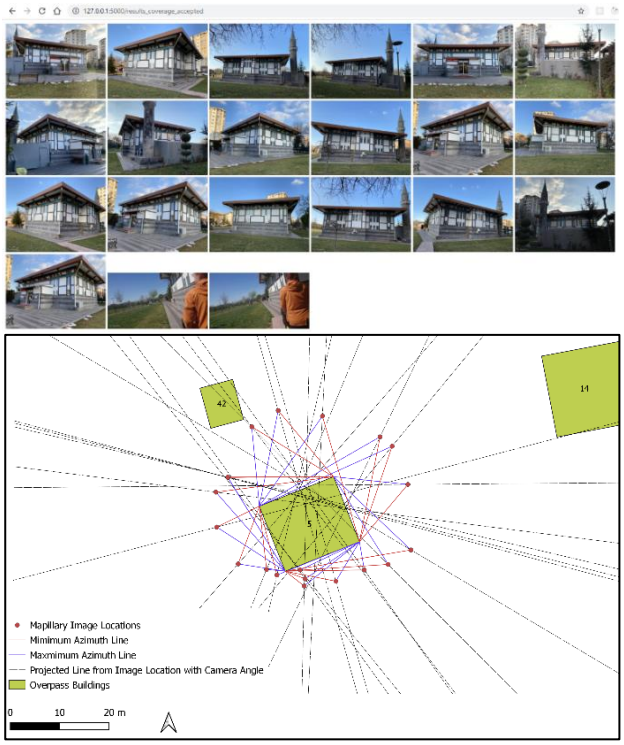

Figure 18. Final coverage accepted images with 20% coverage threshold and locations of coverage accepted images relative to target building 5

Another evaluation was made based on the U-Net segmentation result. The images containing building less than 20% building coverage discarded to be able to find images with higher content of the target building (Figure 18). UNET model is able to refine images with 0.971 accuracy for the target building 5.

The accuracy of the image location can affect the result, especially for images close to the buildings. The difference between calculated and original coordinates provided by Mapillary API has a difference up to 10m (Figure 19). Since it is not possible in our study to measure the exact coordinates for each image location, the computed coordinates are relied on finding the images of the target buildings. However, some of the image locations reside in overpass polygon data, especially those taken in the corner of the building. Although, it could be also due to the accuracy of OSM data that can vary 5 to 10 meters (Calazans Campelo et al., 2017).

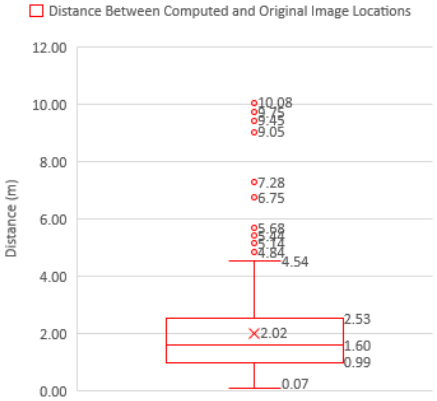

Figure 19. A box plots of distance difference from computed and original image location of 237 images obtained via Mapillary API.

In addition to image location accuracy, the camera angle provided by Mapillary also affected the final result from U-Net segmentation. As the large angle errors affect the correct image selection, smaller camera angle errors can change coverage ratio from segmentation. The image in Figure 20 has cca of 281.22°, ca of 266.12° and approximated angle of 255°.

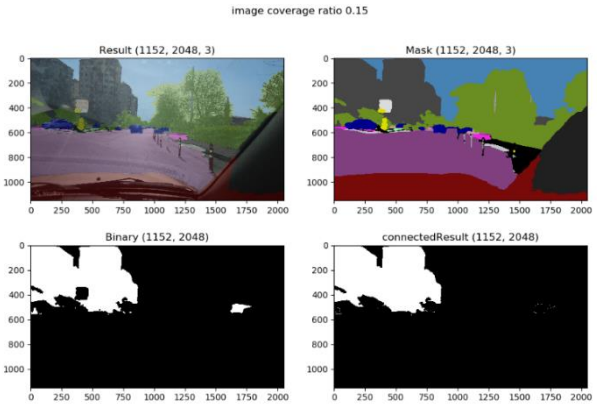

Figure 20. Calculation of incorrect coverage ratio of 0.15 instead of 0. (Image key: 91-rwnEDj3ZQGJlh3pwFPw)

The large camera angle error in cca values also can lead to wrong target selection since the target selection process flow takes only the buildings along with the ± 90° cca value of the query image into the account. In some cases, original camera angles were correct but computed camera angle were incorrectly provided (Table 2) by Mapillary API.

The large camera angle errors in cca should be mitigated in future studies. The images with incorrect cca can have a large discrepancy between ca and cca. Such images can be processed after selecting other images. The histogram matching or other methods can also be used for quick image comparison. Another option, all images can be compared to the closest images within its sequence. If camera angles do not vary with the large value

among the image sequence, the interpolation of the camera angle can be used. This may not work if the sequence is not in order.

Tilted images and images are taken upside down can have incorrect results in the calculation of building coverage ratio as the relative target location in the image (right or left) will be incorrect.

5. CONCLUSION AND FURTHER STUDIES

In this study, it is aimed to automate the image selection process for a target building from Mapillary images though a web application where the user only initiates one image of the target building as a query. Mapillary API and Overpass API were employed to automate selecting street view images from a query image that contains the target building. The developed flask web application runs with the Docker architecture containing a PostGIS database to store tables and to process. Once the user selects a query image from street sequences, the system selects all images visible to the target building of the query image. The first evaluation is based on the image location and camera angle properties, then, the system filters all candidate images seeing the target building partially or fully. Lastly, the pre-trained U-NET model segments the candidate images based on their content, and the building coverage ratio is calculated to select images with higher pixel content of the target building.

In some cases, the camera view angle threshold was too large, and false positive images were returned. Decreasing this threshold was also increased false negative images, therefore finding an optimum threshold based on camera make properties should be investigated. The camera view angle thresholds should be determined differently based on camera type. As a future study, the model will be tested with adjacent buildings in an area with higher building density. The large camera angle errors in cca will be mitigated in future studies, as well.

With the correct camera angle and coordinates, the applied method can successfully retrieve images of the target. For the method to deliver correct images of the target to the user, those images with incorrect camera angles should be identified. The images with incorrect cca can have a large discrepancy between ca and cca. Such images can be processed after selecting other images. The histogram matching or other methods can also be used for quick image comparison. Another option, all images can be compared to the closest images within its sequence. If camera angles do not vary with the large value among the image sequence, the interpolation of the camera angle can be used. This may not work if the sequence is not in order. All mentioned possible options for mitigating wrong cca values will be investigated.

All accepted and rejected images for TB:5 with Th:50° with U-NET can be accessed with their partial EXIF data in the GitHub repository at https://github.com/celikn/SCA_2020.git

REFERENCES

Araujo, A. A., Sampaio, J. C., Evangelista, R. S., Mantuan, A. B., & Fernandes, L. A. F. (2015). Accurate Location of Façades of Interest in Street View Panoramic Sequences. *2015 28th SIBGRAPI Conference on Graphics, Patterns and Images*, 281–288. https://doi.org/10.1109/SIBGRAPI.2015.32

Bulò, S. R., Neuhold, G., & Kontschieder, P. (2017). Loss Max-Pooling for Semantic Image Segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7082–7091. https://doi.org/10.1109/CVPR.2017.749

Calazans Campelo, C. E., Bertolotto, M., & Corcoran, P. (2017). *Volunteered geographic information and the future of geospatial data*. IGI Global.

Cheng, L., Yuan, Y., Xia, N., Chen, S., Chen, Y., Yang, K., Ma, L., & Li, M. (2018). Crowd-sourced pictures geo-localization method based on street view images and 3D reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, *141*, 72–85. https://doi.org/10.1016/j.isprsjprs.2018.04.006

Humbarwadi, S. (2020). *srihari-humbarwadi/Keras-Unet: Keras Unet model trained on Mapillary Vistas*. Keras-Unet, GitHub Repository. https://github.com/srihari-humbarwadi/Keras-Unet

Krylov, V. A., & Dahyot, R. (2018). Object geolocation from crowdsourced street level imagery. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 79–83.

Leon, L. F. A., & Quinn, S. (2019). The value of crowdsourced street-level imagery: Examining the shifting property regimes of OpenStreetCam and Mapillary. *GeoJournal*, *84*(2), 395–414.
Lorenzon, O. (2019). *LatLon, computedLatLon and originalLatLon explained*. https://bl.ocks.org/oscarlorentzon/16946cb9eedfad2a64669cb1121e6c75

Mapillary. (2019). *Mapillary API Documentation*. https://www.mapillary.com/developer/api-documentation/
MapillaryJS. (2020). *MapillaryJS Documentation*. https://mapillary.github.io/mapillary-js/

Neuhold, G., Ollmann, T., Bulo, S. R., & Kontschieder, P. (2017). The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. *2017 IEEE International Conference on Computer Vision (ICCV)*, 5000–5009. https://doi.org/10.1109/ICCV.2017.534

Overpass API. (2020). *Overpass API User's Manual*. https://dev.overpass-api.de/overpass-doc/en/

Panasonic. (2020). *How Focal Length Affects Viewing Angle | Digital Camera Know-Hows | Digital Camera | Digital AV | Support | Panasonic Global*. https://av.jpn.support.panasonic.com/support/global/cs/dsc/knowhow/knowhow12.html

Porzi, L., Bulo, S. R., Colovic, A., & Kontschieder, P. (2019). *Seamless Scene Segmentation*. 8277–8286. https://openaccess.thecvf.com/content_CVPR_2019/html/Porzi_Seamless_Scene_Segmentation_CVPR_2019_paper.html

Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., & Jagersand, M. (2018). RTSeg: Real-Time Semantic Segmentation Comparative Study. *2018 25th IEEE International Conference on Image Processing (ICIP)*, 1603–1607. https://doi.org/10.1109/ICIP.2018.8451495

Wolff, M., Collins, R. T., & Liu, Y. (2016). Regularity-driven facade matching between aerial and street views. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1591–1600.