Scientific Research Publishing

Journal of Geographic Information System, 2020, 12, 620-645
https://www.scirp.org/journal/jgis
ISSN Online: 2151-1969
ISSN Print: 2151-1950

Camera Pose Estimation Using Collaborative Databases and Single Building Image

Bernard Semaan[1,2], Myriam Servières[1], Guillaume Moreau[1], Bilal Chebaro[2]

[1]Ambiances, Architectures, Urbanités Laboratory, École Centrale Nantes, Nantes, France
[2]Lebanese University, Beirut, Lebanon
Email: semaanbernard@gmail.com, myriam.servieres@ec-nantes.fr, guillaume.moreau@ec-nantes.fr, bchebaro@ul.edu.lb

How to cite this paper: Semaan, B., Servières, M., Moreau, G. and Chebaro, B. (2020) Camera Pose Estimation Using Collaborative Databases and Single Building Image. *Journal of Geographic Information System*, **12**, 620-645.
https://doi.org/10.4236/jgis.2020.126036

Received: September 29, 2020
Accepted: November 27, 2020
Published: November 30, 2020

Abstract

Cities are in constant change and city managers aim to keep an updated digital model of the city for city governance. There are a lot of images uploaded daily on image sharing platforms (as "Flickr", "Twitter", etc.). These images feature a rough localization and no orientation information. Nevertheless, they can help to populate an active collaborative database of street images usable to maintain a city 3D model, but their localization and orientation need to be known. Based on these images, we propose the Data Gathering system for image Pose Estimation (DGPE) that helps to find the pose (position and orientation) of the camera used to shoot them with better accuracy than the sole GPS localization that may be embedded in the image header. DGPE uses both visual and semantic information, existing in a single image processed by a fully automatic chain composed of three main layers: Data retrieval and preprocessing layer, Features extraction layer, Decision Making layer. In this article, we present the whole system details and compare its detection results with a state of the art method. Finally, we show the obtained localization, and often orientation results, combining both semantic and visual information processing on 47 images. Our multilayer system succeeds in 26% of our test cases in finding a better localization and orientation of the original photo. This is achieved by using only the image content and associated metadata. The use of semantic information found on social media such as comments, hash tags, etc. has doubled the success rate to 59%. It has reduced the search area and thus made the visual search more accurate.

Keywords

Pose Recognition, Building Detection, Single Image, 2D Map, Collaborative Cartography, Social Media

1. Introduction

Nowadays, city managers are making great use of their city digital representation, and thus have a strong requirement to keep it up to date. Such representation is used for city governance, such as planning, analysis, taxation, security, and many other purposes according to [1]. Beyond this, there exists an Open Data trend, which consists of releasing city information to the general public. For example, some cities, like New York City, give access to some geographical data that is updated daily on [2]. The Inspire European directive [3] also gives a framework to share geographical data in Europe, and cities, such as Lyon in France, are following it and give access to their urban geographical data through an open data portal [4]. Geographical Information Systems (GIS) updates require staff and equipment. Some states, cities, or private companies use aerial images, and some others perform scans of the ground. Both techniques require costs and efforts.

On the other hand, citizens are now equipped with electronic devices capable of taking pictures, videos, GPS locations and other interesting data they can share with other people. In 2015, 58% of the French population had smartphones [5], and some recent statistics like [6] show that the number increased from 67% in 2015 to 77% in late 2016 in the US. The World Wide Web has been evolving for the last decades. Some websites were created to connect people (e.g. Facebook) but quickly turned out to be images, events, and geographical information sharing platform, according to [7]'s numbers. Some other websites were created to share images between photographers or to create a digital portfolio. Finally, websites like Openstreetmap.org provided a platform to add or modify geographical information like roads, buildings, store names, and even benches and trees. This is part of collaborative mapping approaches, which are becoming increasingly available since the Volunteered geographic information (VGI) movement led by [8]. Lately, Google provided a platform for sharing images and geographical information and giving rewards in return to encourage people to participate in its database enrichment [9].

In this article, we propose to use data from these active sources to update the city's information. Publicly shared images may be an interesting source of information. Unfortunately, their localization is not sufficiently accurate: according to [10], the average error of a public GNSS sensor is 5.3 meters under the open sky and this margin of error increases in urban environments due to the signal reflections on the concrete and metallic structures. Also, those images do not include any orientation information in their metadata. DGPE aims to gather these images and try to refine their geolocalization and find the image orientation. We use the image extracted features and its metadata to cross them with other geographical information to provide a refined camera pose. The city's GIS managers, provided with a metric image localization and an orientation, will be able to use this information to start an investigation and update the city's 3D model or a street view database in case of detected changes.

This paper is divided as follows. We first present the related works in the geolocalization field based on single images. We then present DGPE, and detail every layer. We describe the building detection method we have called Segments Based Building Detection (SBBD) and show its results in many image processing challenges. Finally, we present some results and case studies of DGPE, we show some statistics for two image databases, and conclude.

2. Related Work

Image geolocalization is a long and complex process that consists of finding the exact image location on earth. This location, defined by latitude, longitude and altitude, should be extended with orientation, tilt and roll angles from the camera parameters to be able to determine the camera pose, and thus what an image shows. Therefore, image geolocalization research works have been carried out on two main scales and thus: "Image Geolocalization" that returns a wide geolocalization (e.g. A country, a city or the type of nature where the image was shot) like [11] and [12], and "Pose Recognition" that limits the search to a reduced area where we can use more precise reference data in order to find an exact geolocalization and camera orientation, as for [13] [14] [15] [16] and [17].

We present in the first part of this section a few image geolocalization approaches as well as some pose recognition approaches, we then present some building detection methods using image processing that then will help us to refine the pose finding a correlation between the image and GIS information.

2.1. Image Geolocalization and Pose Recognition Using Geometric and Semantic Information

The authors of [11] present a system that runs 24/7 to automatically download images and extract visual knowledge from the Internet data. The system discovers objects, scenes, and common-sense relationships in the downloaded images. The system aims to annotate an image with minimum human labeling effort. This approach helps in some cases finding a geographical location out of the image content, for example, a leaning tower refers to Pisa in Italy, a pyramid refers to Egypt, etc.

Another research, [12], uses only the image textures descriptors and deep learning to find its geolocalization. The authors mention that it seems "exceptionally difficult" to find an image location using only the textures' information. However, the authors use image content such as landmarks, weather patterns, vegetation, road markings and architectural details to accomplish the task. Only 14.9% of images have accurate geolocalization to the scale of a city street (1 km), 20.3% to a city-scale (25 km), and the rest is worse.

Long-short term memory (LSTM) architectures [18] uses multiple images from the same photo album and pushes the results to 32% of successful geolocalization at the street scale and 42.1% at the city scale. However, the image geolocalization presented above is not accurate enough for GIS updates and changes detection in a city, since they are still considered as a wide-scale geolocalization, instead of precise pose estimation. The geolocalization accuracy of such techniques is not sufficient for our research purposes, yet it still gives an idea about where the image was shot.

In the article [13] the authors consider that automatic registration by coupling a textured 3D GIS model and 2D images is an efficient way of identifying buildings in an image or a video stream. They start their work assuming that images with a valid orientation and GPS location are already registered in the database. The paper considers extracting for every building facade in the registered images a corresponding texture descriptor. Then, to register a new image, it uses SIFT descriptors by [14], matching with the registered texture descriptors. The limitation of this work lies in affine rotation (30 degrees or more) and building facades should be very different from one another to avoid SIFT descriptors confusion.

The algorithm used in [19] achieves fast and accurate feature landmark-based camera parameter estimation by reducing the number of matching candidates and assigning priorities to them. They aim to find camera pose by matching the image with known landmarks that have been previously accurately scanned using a laser range sensor. Therefore, a 3D model landmarks database is unavoidable with such a technique.

The authors of [20] propose a combination of simultaneous localization and mapping, called SLAM, and a global localization method. The authors consider an AR application built on smartphone devices that will use the first two frames to initialize a structure from motion reconstruction. Then, the server will then compare the reconstructed environment with a global cloud of points. Finally, the client will update the current pose using his sensors and the initial information.

On the other hand, Bioret *et al.* [16] propose a localization approach in urban environments based on correspondences between 2D GIS and a single 2D street photograph. Practically, Bioret uses the vanishing points and the limits of the facades in the 2D image to find the angles between facades and their width ratio in the image building. The authors use this information to query a GIS and get the relevant pose. Finally, we note that the algorithm limits the search zone to a 100 m range around an initialization point and is not fully automated.

Lastly, a similar approach [17] that matches 2D GIS with 2D images uses an automatic building detection technique. They automatically detect the vanishing points, then find vertical segments that should represent building corners. They then use geometric correspondences between previously extracted features and a 2D map to find the pose that is the closest to the initial GPS information.

Urban images contain geometric information that describes building shapes, and texture descriptors. Additionally, semantic information, such as storefronts, street names, and other detectable text, can reveal localization information.

In [21], the authors try to find stores using city images. The system extracts the text from the image and compares it with the stores' names surrounding the image localization using lists like [22] and [23].

2.2. Building Detection and 3D Scenes Reconstruction

In this section, we present some building detection methods that have been developed by other researchers as we will use building detected in the image and GIS information for pose image refining.

Urban environment understanding is an active research field partly based on street images use. Several techniques like [24] [25] and [26] used stereography to reconstruct urban environments using several images or video sequences. However, we seek to reconstruct an urban environment, or at least understand it, using a single 2D image. Therefore, the preceding methods are not suitable for our image datasets. Derek Hoiem *et al.* [27] [28] propose a method of 3D environments reconstruction using 2D images. They divide their images into superpixels and estimate the orientation of each one. A superpixel is a group of pixels that are close to each other and have similarities in their features, see [29] for more details. They annotate the image superpixels to three main categories: ground plane, surface sticks up from the ground or sky. The second category is then divided into four sub-categories: surfaces facing left, right, or toward the camera and non-planar surface like vegetation. The authors do not aim at understanding the image content but at retrieving its orientation. In [30], the authors try to estimate the depth of each pixel from a monocular image. They proceed by labeling pixels using what they call semantic segmentation of the scene. They consider that a pixel labeled as sky should be far away, another pixel label as ground is horizontal, etc. This method may be used for depth estimation and image segmentation as a prior step of building shape detection. Besides, [31] uses statistical learning algorithms to label pixels according to three categories: vertical, ground, and sky. The algorithms are already trained using other urban images. The algorithm finds the vertical elements in an image, including buildings, but does not find the building shape.

Lastly, [17] tries to find the building corners in order to compare them with a 2D map. They start by computing the Tilt-Invariant Corner Edge Position (TICEP) features by sequentially applying vanishing point estimation, corner edge identification, and tilt angle normalization. An important part of the TICEP detection is normalizing the tilt angle. For this purpose, they estimate the camera parameters, like rotation and focal length, to perform the correction of the vertical vanishing point. Therefore, this last method can only detect the building boundaries from two sides. In other words, only 3 building vertical edges can be detected, even if more are available. Available vertical edges can only be transformed into two facades, although [16] proved that when more facades are available, his 2D GIS matching performs much better.

Our goal is to refine the geolocalization of online published photos. Therefore, we only have a single monocular image and rough GPS information with no orientation information. After exploring the state-of-the-art techniques, the ones that fit the most with our datasets should be [16] or [17]. We thus propose in section 3 the Data Gathering system for image Pose Estimation (DGPE) that will help to refine image geolocalization and orientation using a single image, a 2D map, and corresponding techniques, such as [16]. We enrich our process with semantic information that could be retrieved from the social media metadata, text, or landmarks detection in the photo.

3. Data Gathering System for Image Pose Estimation (DGPE)

In this section, we propose the DGPE system for image geolocalization using simple and widely available information. DGPE takes as input 2D images, either from a dataset we have already shot or images downloaded from social media websites (Flickr, in our case). The second input information is 2D maps that represent the city buildings.

All the images that we will present were taken by us using an iPhone 5 device with a GNSS sensor. To save the reference localization, the photographer's location was pinned on a satellite view map, and the location coordinates were then retrieved. We assume using in DGPE dynamic information sources (updated regularly), therefore we used maps extracted from OpenStreetMap.org. We also used the Nominatim API [32]. This API takes an input one or more keywords. For every keyword, it returns a list of possible localizations. This will help, when using the metadata and the semantic information, to obtain better localization results. The final result of DGPE is a GPS location and a camera orientation information.

When there are several possible results, DGPE aims to provide the best localization results for the city's GIS managers to help them decide whether to update the geographical information or not.

3.1. A Three-Layer System Architecture

We have created a three-layer system and summarized its main functionalities in the diagram shown in **Figure 1**. The first layer of DGPE is the "Data retrieval and preprocessing layer". This layer takes as an input the user requirements about the zone one wants to find new images in. DGPE reads the images from a local dataset or downloads images shared on social media websites and filters them by keeping only the ones verifying the user needs. The "Features extraction layer" is the second part of DGPE. Its function is to extract from every image the existing features in the visual part of the image. We divide the extracted information into two categories: Geometric information that represents the building shape (ratios and angles between facades) and Semantic information that provides information about the image location (stores logos, street names, landmarks, metadata, etc.). Extracted semantic and geometric information will then be compared to other geographical information that will produce a list of camera poses or localization information. The resulting lists will be crossed in the next layer to keep the most relevant ones. The final layer of DGPE, "Decision making",

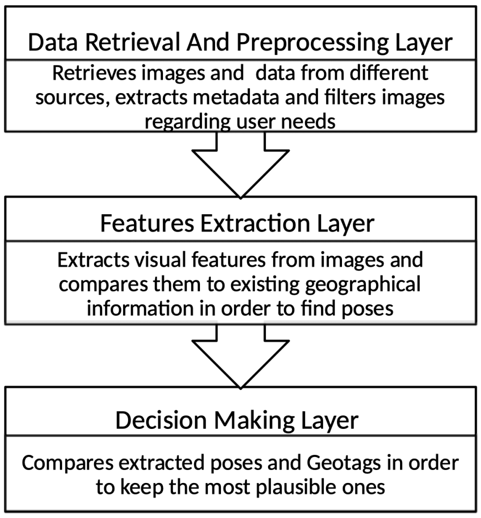

Data Retrieval And Preprocessing Layer
Retrieves images and data from different sources, extracts metadata and filters images regarding user needs
Features Extraction Layer
Extracts visual features from images and compares them to existing geographical information in order to find poses
Decision Making Layer
Compares extracted poses and Geotags in order to keep the most plausible ones

Figure 1. The global DGPE diagram.

helps to reduce the number of solutions from the previous layer. The results from multiple sources will be crossed, and only some will be preserved. A final process will try to validate these poses by comparing them to online published geolocalized street images.

3.2. Data Retrieval and Preprocessing Layer

The first layer of DGPE retrieves the images from local or online databases and filters them using the user requirements. The user should specify at least the search zone he/she wants to extract images from. He may also add a minimum and maximum date limitation in order to avoid retrieving images that have been already processed. We have designed DGPE using multiple modules to allow modifications or module replacement later on. This layer comprises four modules presented in this section.

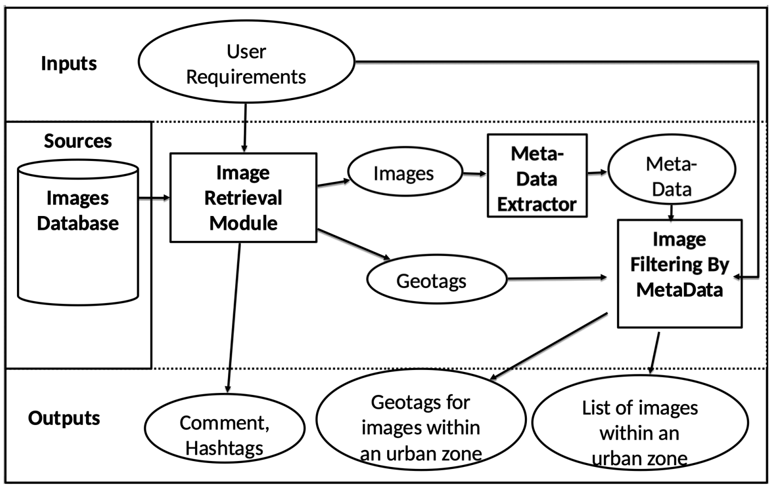
The image retrieval module is responsible for querying images from the image database. Two options are available for images and their metadata retrieval: from a local dataset or an online database. When using the online image retrieval module, many advantages exist. First, retrieved images belong to a restricted geographical zone that is determined beforehand by DGPE user, image shooting or uploading date are also taken into consideration. Therefore, we start by filtering metadata in this case. A second advantage is the metadata information that we can retrieve from such a database. Regardless of the GPS location, the image shooting or uploading date, images retrieved from the social media websites also have some semantical metadata that could be useful for our processing. The module retrieves hashtags, image descriptions and titles that are in plain text format. On the other hand, images retrieved from local datasets or databases that store and allow to retrieve EXIF[1] metadata are sent to the Meta-data Extractor module.

[1]See http://owl.phy.queensu.ca/~phil/exiftool/TagNames/EXIF.html for list (*consulted* 2020-10-17).

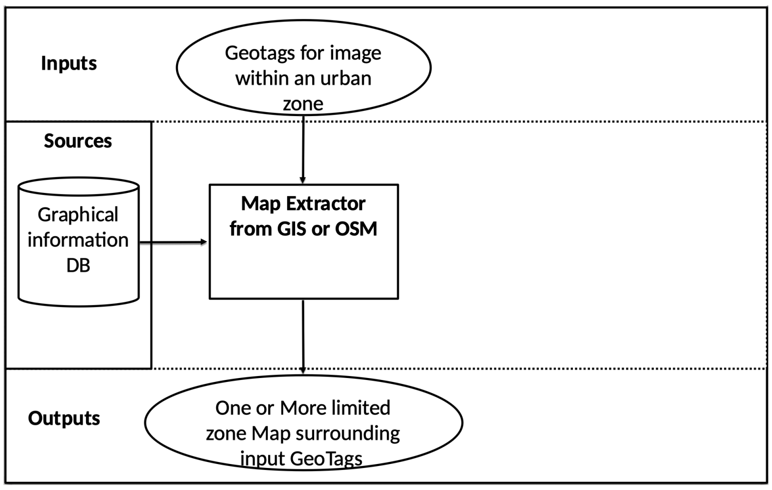
This simple module extracts geographical information and camera shooting parameters to initialize the next steps of the system. These images are also filtered using the image filtering module according to their metadata, especially the location information, to make sure that the image belongs to the user's specified zone. The results of both image retrieval modules are an image and initial position information about the image. When available, a list of words extracted from the metadata information is also returned. Finally, in Figure 2(b), the map extractor module takes the GPS location of the image and extracts from the city map a limited zone of approximately 200 m × 200 m surrounding the initial GPS location. We assume that a 200 meters edge square bypasses the smartphones GNSS sensors average error presented in [33].

3.3. Features Extraction Layer

This layer of DGPE is an important phase of the processing. Image metadata and semantic information extracted in the previous layer are helpful to find the location



(a)


Inputs
Geotags for image within an urban zone
Sources
Graphical information DB
Map Extractor from GIS or OSM
Outputs
One or More limited zone Map surrounding input GeoTags

(b)

Figure 2. (a) Image and metadata retrieval; (b) Zone extraction.

of our image. Yet, finding orientation information and exact camera location remains the main purpose of DGPE. Pose detection can be fulfilled using geometric features matching with the 2D map of the city. First, we explain the semantic analysis part of this layer and then we will explain the geometric part.

3.3.1. Location Detection Using Semantic Information Retrieval

Figure 3 shows a module that finds possible geotags of an image using textual input information. The text can come from the metadata of the image extracted from an online database as described above. Images can also include semantic information in their visual parts, such as landmarks, logos from storefronts, urban furniture and text. We have applied the text recognition extraction because it automatically returns the textual information visible in the image. Store logos and city landmarks may also be useful in such analysis, we are looking forward to including them in future implementations. Several text recognition techniques have been developed in previous work [34] [35] [36]. In our implementation, we have used Google Vision API, but it could be replaced by any other text detection method in an image.

Another semantic aspect of our images is the content that they are showing. Images within an urban zone can be taken on the streets and represent buildings, but others may represent people, food, or indoor locations. We thus use the Google Vision API to make sure the image we are using in DGPE does contain buildings. The API returns a list of words describing the scene for every image. We only keep images that have in their description list words related to the buildings (e.g. Building, architecture, property, facade, town, house, etc.), and other images are skipped by the system. The list of words was chosen based on statistics performed using responses returned from the same API. The selected words are the ones that had a confidence score above 90%, given by the API.

Lastly, a list of textual information is created using the words detected from the visual part of the image, as well as the text retrieved from the image's metadata found on the image-sharing platform. The list is then passed into

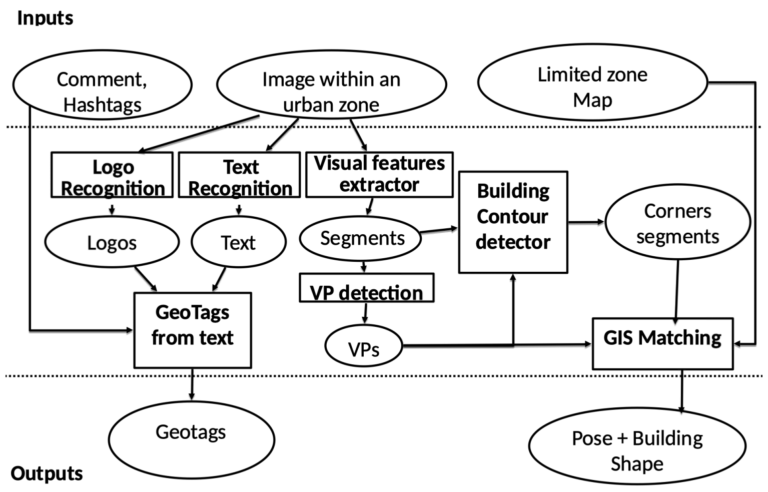


Figure 3. Features extraction layer.

a new module that will find geotags using the textual information. As already presented in the general overview of DGPE, we use Nominatim API for this step. The search zone is restricted to the limited zone previously configured. We explain in the next layer the way we use this geographical information.

3.3.2. Location Detection Using Geometric Information Retrieval

We assume in this part of DGPE that a building can be represented using segments and simple rectangular polygons. We will now explain the global geometric process for image geolocalization. We will, later on, explain in detail the building detection module developed in this part of DGPE. We can find in **Figure 3** a module named visual features extractor. It aims to extract basic features from an urban image. We thus search in the features' extraction module to find all the segments that could be extracted from the image. For this, we use the LSD [37] segments extractor. It returns several segments of different sizes. Small segments are filtered out using a parameter relative to the image size. These segments are usually found in far buildings, trees, and other urban furniture that is not related to the main building in the picture. The rest of the segments is then used to detect vanishing points and the buildings' outline. Three vanishing points are detected in the image using the [38] technique. Lastly, the detected building facades and vanishing points are transferred to the GIS matching module that will return a list of camera poses. This module uses the technique presented in [16], which compares the angle between facades, as well as their length ratio, to find possible poses in the restricted zone map previously extracted.

3.3.3. Building detection process

We now explain our building detection method, which performs better in DGPE than the state-of-the-art methods. This method will be used during the GIS matching process mentioned in section 3.3.2.

1) Segments Based Building Detection (SBBD)

In the following, we present our Segments Based Building Detection (SBBD) method using several algorithms that complete each other in order to detect the building's facades. SBBD only uses the segments found with the LSD algorithm [37], as well as the detected vanishing points. The result of SBBD is a group of facades, each presented by two vertical segments and a horizontal vanishing point. One vertical vanishing point is considered for every processed image.

2) Segments chaining algorithm

We now present the strategy used to regroup small segments detected using LSD [37] and the construction process of bigger ones that are more relevant to our building envelope detection.

LSD segment detection will provide us with a list of segments of different sizes (**Figure 4(a)**). We have first chosen to limit the number of tiny segments to avoid high calculation cost. In fact, tiny segments in an urban street image are usually found in trees, clouds, and some urban furniture. A parameter relative to the image size was experimentally chosen to filter those segments.

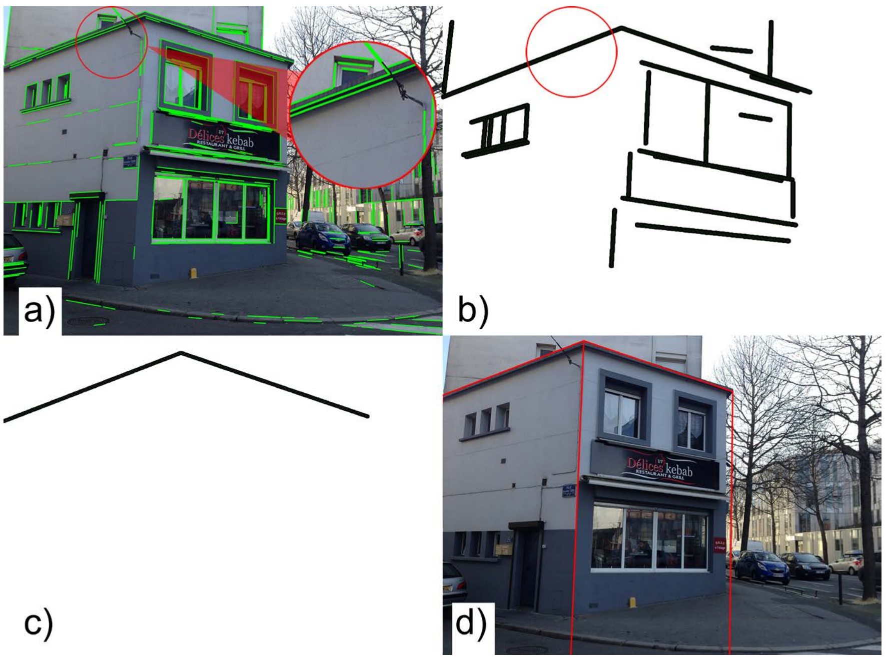


Figure 4. Building contour detection.

We then filter segments by elevation from the ground; only the highest segments in the image are important for SBBD. We thus discard the segments that are closer to the ground to avoid more computation and filter segments related to cars and pedestrians.

Once filtering is done, we compare every two segments among the remaining ones and regroup them until convergence. To group two segments, two parameters are previously defined: the angle in between when the segments extensions intersect, and the minimum distance that separates the segments. If the angle between the two segments' intersection is smaller than the one defined in SBBD settings (Figure 5(b)), and the distance separating the two segments edges is smaller than the maximum distance specified in SBBD parameters (Figure 5(a), Figure 5(b)), the segments could be grouped. Otherwise, the segments remain separate and processed in the next steps (Figure 5(c), Figure 5(d)). We can find two cases when regrouping two segments. The first case is when the segments chaining result is equal to one of the input segments, we thus keep that one and remove the other segment from the list (Figure 5(e)). The second case is when the segments chaining result is bigger than both segments, we thus add the segment to the list and remove both original segments (Figure 5(a), Figure 5(b)). The final result looks like Figure 4(b).

3) Building envelope segments detection

We explain in the following, the way we filter the segments to keep only the ones that represent the building envelope segments. The segments are compared two by two until convergence. Every two segments are compared to test if they are on top of each other, and when it is the case we only keep the highest segment.

We first find the highest point of every segment to determine the highest segment. We then compare the distance that separates the segments' endpoints. We use a tolerance value to avoid removing segments overlapping by a small number of pixels (see example in Figure 6(b)).

The next step consists of finding if at least one segment endpoint of *s*1 is between the two segment endpoints of *s*2. This is done by comparing their horizontal projection *x*. In that case, the segments overlap, the distance of overlapping is bigger than the tolerance one (see Figure 6(c)), SBBD keeps only the higher segment.

Finally, when the segments do not overlap (see Figure 6(a)), SBBD keeps both segments. The final result is presented in Figure 4(c).

4) Finding missing building envelop segments

We finally illustrate the algorithm used to add missing segments to the building envelope and the vertical corners in Figure 7. We start by finding for each horizontal segment's extension, the intersection it makes with the next segment. If the intersection point is between the two segments' endpoints, we extend both segments until their intersection point, see the example in Figure 7(a). If the intersection point does not belong to the interval between the two segments'

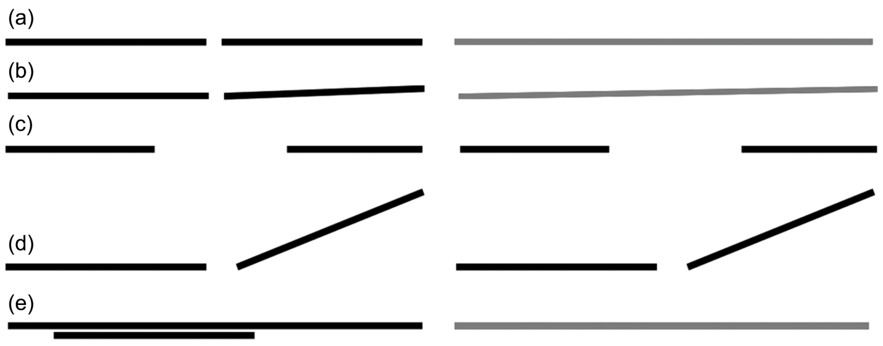


Figure 5. Grouping possibilities.

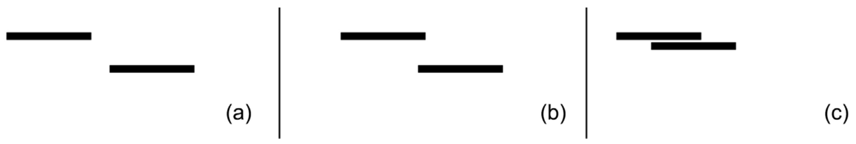


Figure 6. Segments overlapping possibilities.

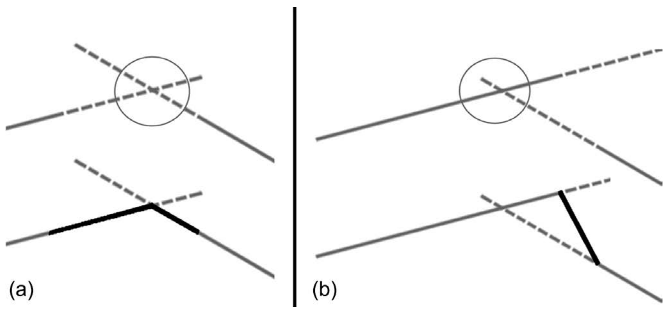


Figure 7. Add missing segments cases.

endpoints, we add a new segment that links the two segments' endpoints (**Figure 7(b)**). We then reduce the segments into single points and remove the ones that are too close to avoid detecting tiny facades. The points are then transformed back to segments and a vertical edge is added on every building corner (**Figure 4(d)**). Every vertical segment should intersect with the bottom of the image and pass by a building corner and the vertical vanishing point.

3.4. Decision Making Layer

In this layer, we explain the applied filtering process to keep only the most plausible camera poses of the image.

Figure 8 shows the module that composes this layer. It compares two lists of geographical information: 1) the GPS locations, also called geotags, that have been found from the previous layer using semantic analysis plus the initial GPS location of the image, and 2) the poses that have been detected by matching the building shape with the 2D GIS described previously in paragraph *Location detection using geometric information retrieval*. The module uses the geotags to filter the poses found from the geometrical matching. When common angles are presented, 90 degrees building corner for example or continuous buildings block with no clear buildings' limitations, the GIS matching process returns a large number of solutions. We filter those results by taking around every geotag from the list, all the camera poses that exist in a 20-meter radius. Few poses remain after this process.

We have chosen to compare the initial image with other images retrieved from GoogleStreetView in order to match the visual features. Some techniques, like [39], use GoogleStreetView to detect a camera pose of the image. However, GoogleStreetView images, when available, are shot every five to ten meters on a straight line. GoogleStreetView matching technique results cannot be as accurate as building shape identification poses like [16]. More possibilities are available in GIS matching techniques. Therefore, we download the images from GoogleStreetView using the remaining poses. We then compare them with the original image using SIFT descriptors matching. This process will add trust in the pose we have found using the GIS matching process. On the other hand, this step

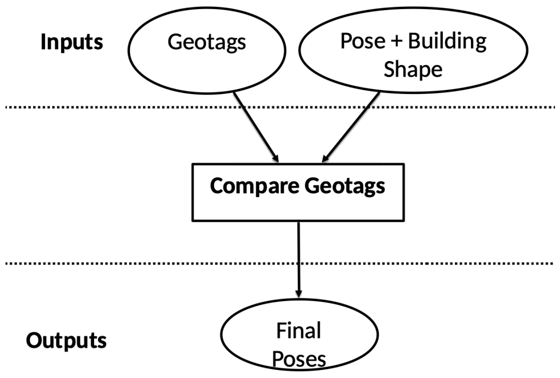


Figure 8. Decision making layer.

does not confirm a pose validity and is not used to find the image pose by comparing it with GoogleStreetView images. Finally, the layer returns to the user one or more likely poses of the camera with a confidence score based on the number of correspondence points with the GoogleStreetView image when available.

4. Results

In this section, we present some building detection results using our method. We compare our method in different weather conditions and prove its robustness against weather changes, shadows, and some occlusion problems. We then compare our method with the one described in [17], using our datasets and theirs. Finally, we show a case study that explains the evolution of our processing chain.

4.1. SBBD Results

We have tested SBBD using different conditions and datasets. In **Figure 9**, we have tested the method on urban images under two different weather conditions. In the top row of **Figure 9**, the images were taken in January 2017 on a sunny day. Trees were smaller and some with no leaves, some buildings were exposed to the sun and its shade. In the bottom row of **Figure 9**, images were taken in April 2017 on a rainy day. Tree leaves started appearing, announcing the spring, some other trees had grown and in some images, scaffolding was added to the building for renovation. Yet, we have obtained the same detection of the building. Even if in some cases it was not a perfect detection of the complete building, it gave us the same information about the facades and the angle in between.

We have then summarized SBBD results in **Figure 10** chart. Chu *et al.* detection results are presented in red, and our method detection results are presented in blue. Chu *et al.* dataset consists of 252 building images[2] captured each from a different point of views to represent eight buildings' edges. Those images were screen captured from spherical images uploaded to GoogleStreetView. Our dataset consists of a single image of 19 simple buildings taken from a pedestrian point of view.

We have run the building detection algorithm on the available photos using both methods. Chu *et al.* method's aims to detect three vertical building edges. Those edges draw the limits of two facades of the central building in the image. We thus consider a detection is successful when both facades are found, otherwise the detection is considered as a failure.

SBBD's goal is to find a facade or part of a facade by finding both sides' vertical edges. Since a single facade is not sufficiently relevant to find a building in the GIS, we aim to find at least two facades in the image. We thus consider a successful detection when two or more facades are successfully detected. We consider a wrong detection when a single facade or no facades were detected.

[2]Images downloaded at https://github.com/chuhang/GPS_Refinement/tree/master/dataset (*consulted on* 2020-10-17).

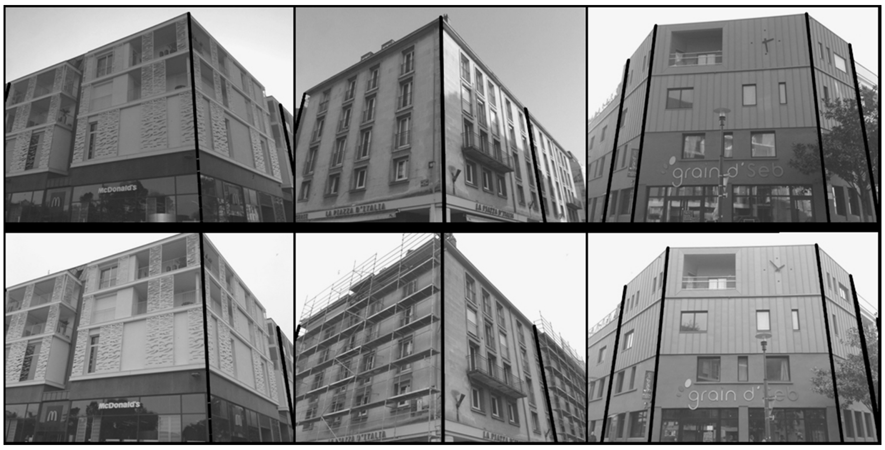


Figure 9. Building contour detection in good and bad weather.

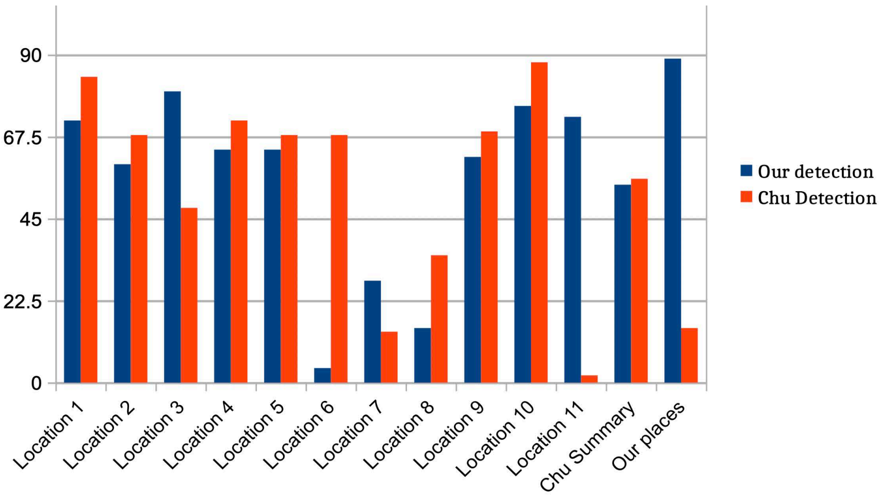


Figure 10. Building detection summary chart. Ordinate axis shows the rate of good building detection.

Regarding the Chu dataset, we have found that his method gives 56% (see Figure 10 Chu Summary column) as a whole. We have also tested Chu's dataset using SBBD that gave slightly lower detection performance with a 54% detection rate (see Figure 10 Chu Summary column). We have found that when our method works on a specific location, no matter how the camera moves, the method can still detect the building. Yet, the detection fails when the roof edge includes a lot of small details (Location 3 in Figure 11), or when many buildings exist in the image (Locations 6, 7, and 8 in Figure 11).

We have then tested our dataset with both building detection methods. We have found that on our dataset, SBBD performs better than the Chu *et al.* one. In fact, Chu's method found only 15% (see Figure 10 Our places column) of the buildings in our dataset. On the other hand, we used SBBD and obtained a successful detection rate of 89% (see Figure 10 Our places column).

We have thus chosen some photos to show the building detection results of both implementations in Figure 12 and Figure 13. Results of SBBD were good

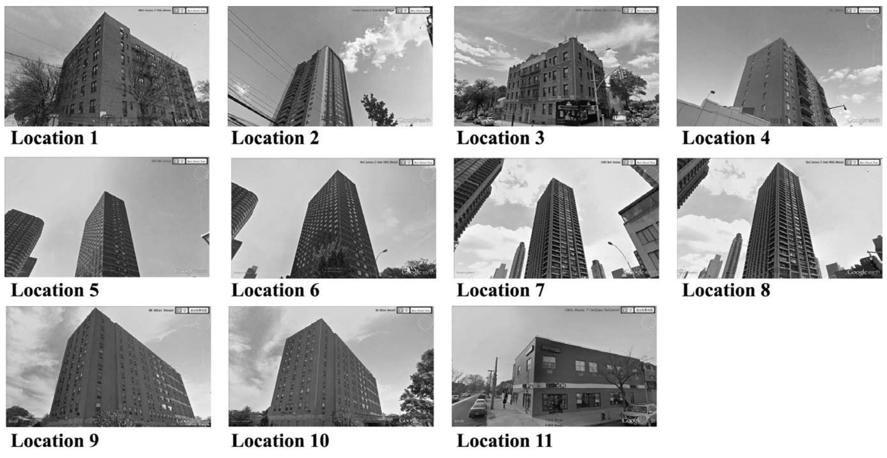


Figure 11. Snapshot of Chu's 11 locations dataset presenting 8 different buildings.

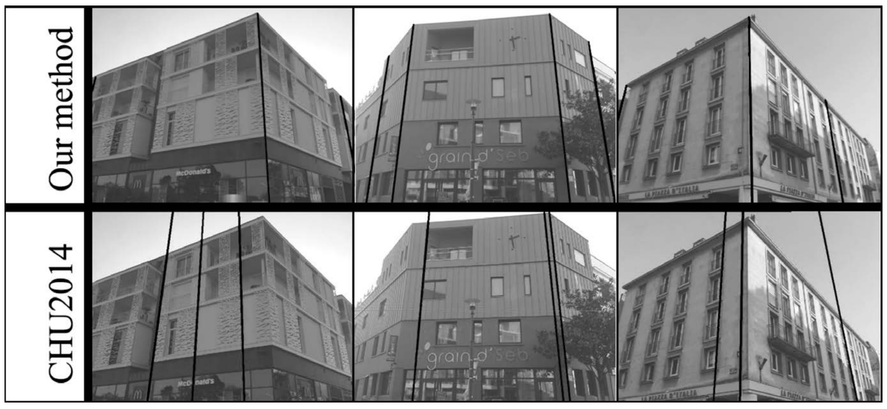


Figure 12. Building contour detection comparison with [17] on our datasets.

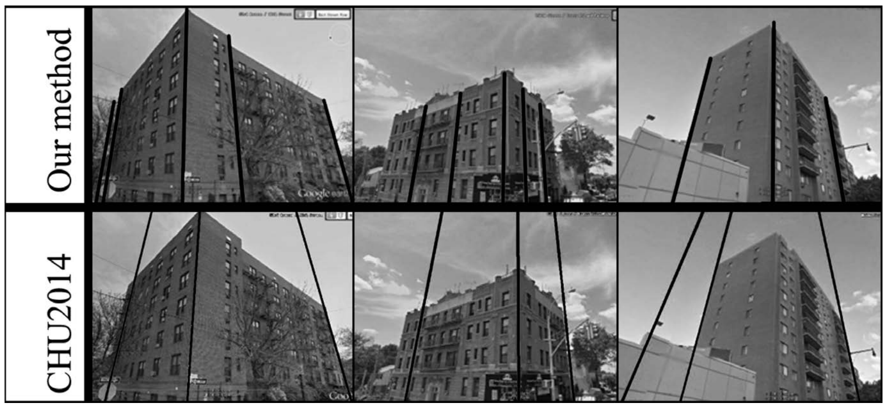


Figure 13. Building contour detection comparison with [17] on their datasets.

so far, complex buildings were challenging but the results we have found are encouraging. In some cases, we find better results than the ones returned by Chu *et al.*'s method. Beside, SBBD was able to detect more than two facades, which could help matching better a building with a 2D map.

Finally, SBBD has its limitations too. Tilted roofs, inclined walls, complex architectures, a single wall with several facades and important occlusion problems cause detection fails. We also note that SBBD detects only a single building in the image, multiple buildings in the same image are thus badly detected. Some wrong detection results are presented in Figure 14.

4.2. DGPE Results

We present in this section two case studies that summarize the main steps for image pose recognition with DGPE. We show some image pose recognition using automatically extracted building shape and 2D GIS matching and some semantic information. We compare the image pose found using DGPE with the reference pose we saved during the image acquisition using the satellite view.

4.2.1. First Case Study

Figure 15(a) shows the image whose pose we would like to find and the rough localization found in its metadata, represented with a black pin. No orientation information is available in the image's metadata. In Figure 15(b), an approximate 200 m × 200 m map is visualized with the possible poses found. These poses, drawn in dark gray dots and polygons, resulted from the building detection and 2D GIS matching. Buildings are drawn in light gray. In Figure 15(b) poses are hardly visible because of their huge number, 644 exactly. Black circles with letters represent the zones extracted using semantic information and existing metadata. We have found for this image four locations that will help to filter the 644 poses previously found. We filter all the poses that do not belong to the 20 meters radius circle surrounding the semantic pose.

In Figure 15(c), we show the remaining poses in four reduced maps. The semantic poses (*i.e.* poses extracted from semantic data) are represented using

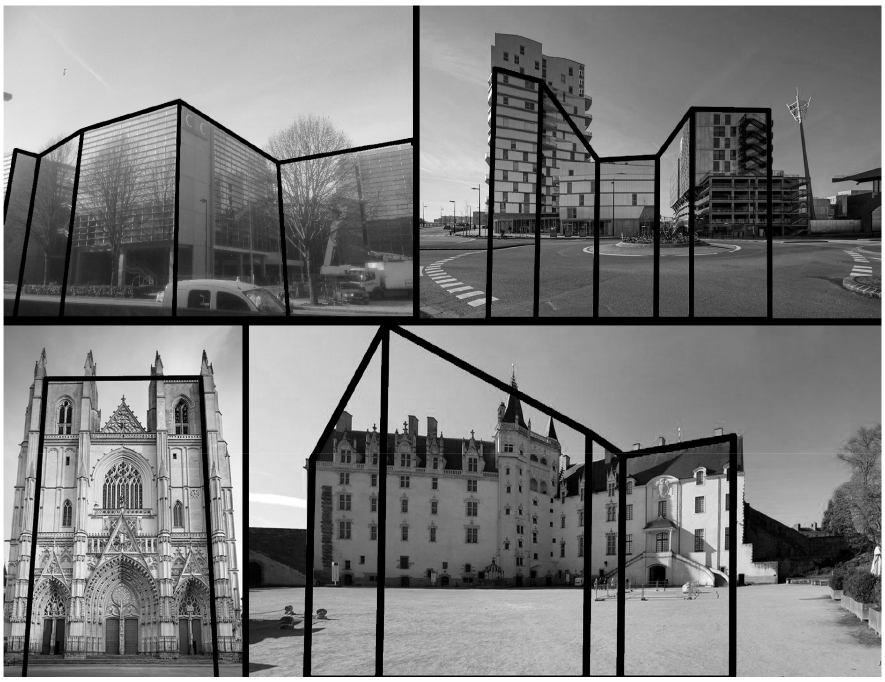


Figure 14. Wrong building contour detection results.

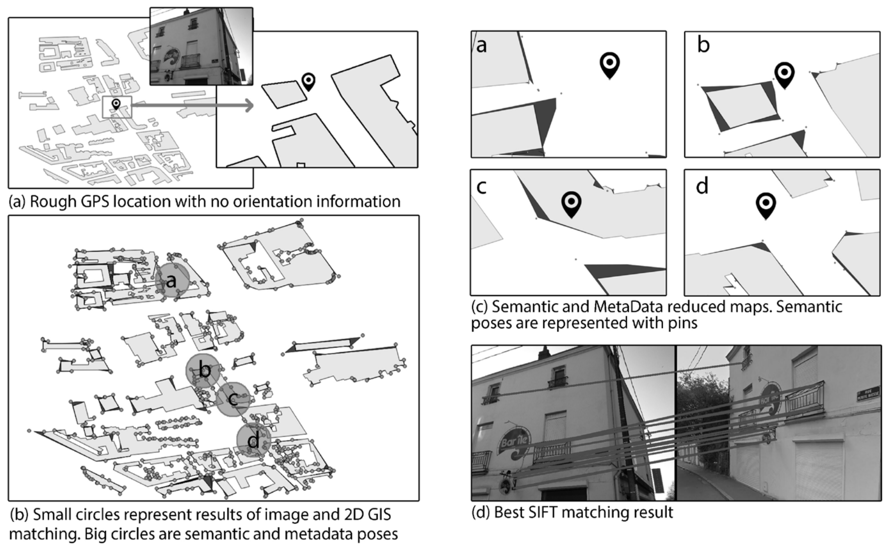

(a) Rough GPS location with no orientation information
a
b
c
d
(b) Small circles represent results of image and 2D GIS matching. Big circles are semantic and metadata poses
(c) Semantic and MetaData reduced maps. Semantic poses are represented with pins
(d) Best SIFT matching result

Figure 15. Image pose recognition process.

black pins, buildings in light gray and the 18 remaining possible poses in dark gray. All poses are used to download a GoogleStreetView image with the corresponding location and orientation, then they are compared with the initial image using SIFT descriptors. The best SIFT matching result is presented in Figure 15(d). Some architectural elements and commercial details are detected in this image. Therefore, we consider the pose having the biggest number of matched SIFT features as the most trustful.

In Figure 16, we show the evolution of the result and the reference data. The black pin shows the initial GPS information found in the image metadata. Black and white and square textured polygon shows the best result returned by DGPE (location and orientation information). The black dots textured polygon represents the reference pose of the image.

We can see that the image location evolved from 7.3 m away from the reference location to a 3.98 m location error. Yet, the most important is the detection of the camera orientation that almost fits with the reference orientation.

4.2.2. Second Case Study

Figure 17(a) shows the image whose pose we would like to find and the rough localization found in its metadata represented with a black pin. No orientation information is available in the image metadata. In Figure 17(b), an approximate 200 m × 200 m map is visualized with the possible poses found. These poses drawn in light gray dots and polygons resulted from the building detection and 2D GIS matching, buildings are drawn in dark gray. In Figure 17(b) poses are again hardly visible because of their important number, 1104 exactly. The black circle represents the zone extracted using semantic information. We have found for this image only one location that will help to filter 1104 poses previously found.

In Figure 17(c), we show the remaining 1104 poses in a reduced map. The semantic pose (*i.e.* pose extracted from semantic data) is represented using a black pin, buildings in dark gray, and the 69 remaining possible poses in light gray. We recall that the remaining poses are the ones that belong to a 20-meter ray around the semantic pose. We download a GoogleStreetView image using every remaining pose's location and orientation. We then compare the downloaded images with the initial image using SIFT descriptors. The best SIFT matching result is presented in Figure 17(d). Some architectural elements and commercial details are detected in this image. Therefore, we consider the pose having a bigger number of matched SIFT features as the most trustful.

In Figure 18, we show the evolution of the result and the reference data. The black pin shows the initial GPS information found in the image metadata. The black and white square textured polygon shows the best result returned by DGPE (location and orientation information). The black dots texture polygon represents the reference pose of the image.

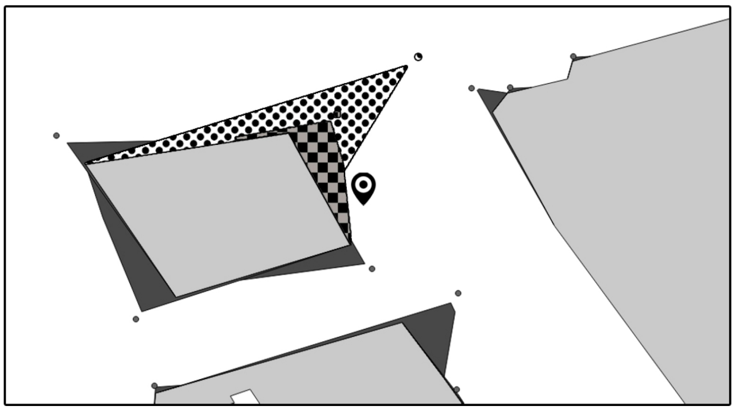
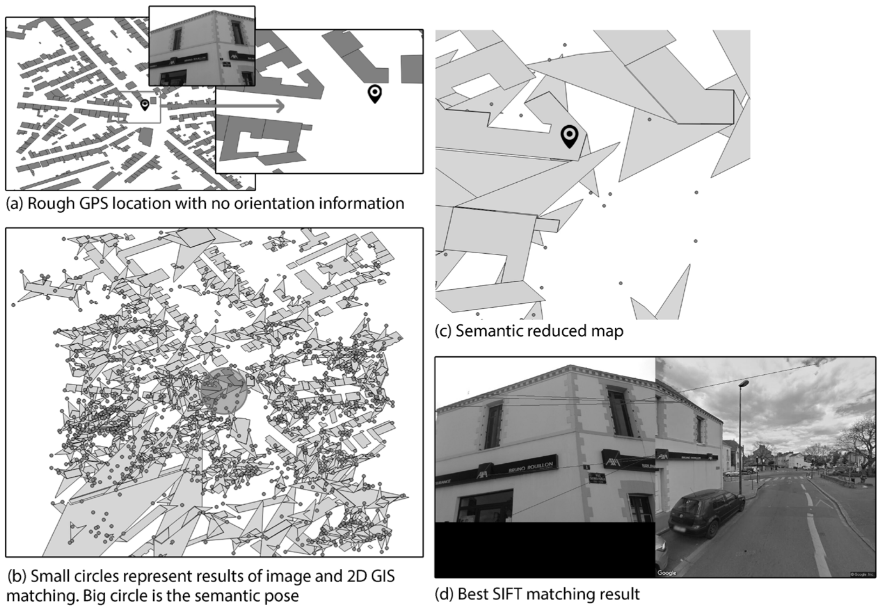


Figure 16. Image pose recognition result.


(a) Rough GPS location with no orientation information
(b) Small circles represent results of image and 2D GIS matching. Big circle is the semantic pose
(c) Semantic reduced map
(d) Best SIFT matching result
Figure 17. Image pose recognition process.

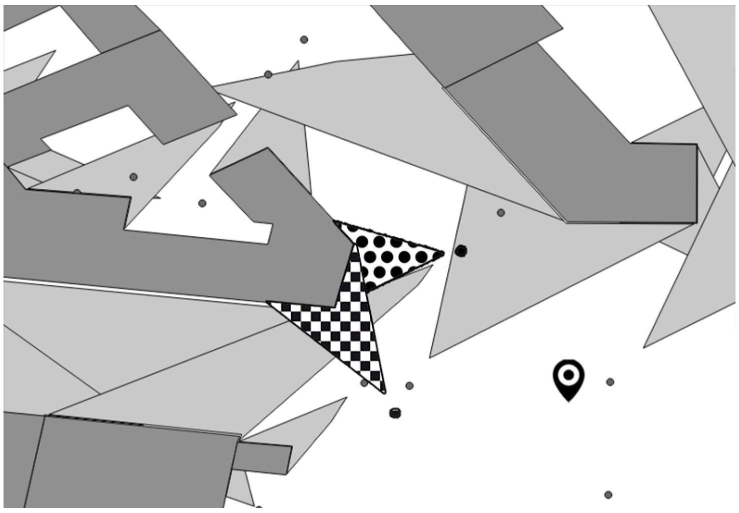


Figure 18. Image pose recognition result.

We can see that the image location evolved from 18.3 m away from the reference location to a 10.8 m location error. However, the most important information found is the camera orientation that shows the same building as for the correct reference orientation and a common facade for both reference and the detected pose.

4.2.3. Numbers and Discussions

We present in this section some charts showing the results of DGPE applied to two image databases we have created. The first one contains 19 images that have been used with no associated semantic data, the results are presented in **Figure 19**. The second database contains 28 images that we have uploaded to the "Flickr" platform and thus contains additional semantic data available as comments, image titles and hashtags. We compare this database's results when using semantic data (**Figure 20**) and when semantic data are ignored (**Figure 21**) in order to understand the importance of this information in the pose detection process.

We can find in **Figure 19** that we detected in 32% of the images part of the text that may reveal geographical information. In 21% of the cases, we find only the text that reveals geographical information and in 21% of the images text that reveals geographical information is detected with useless text in addition. This textual information is then used to find the photos' geolocalization. DGPE finds in 36% of the cases the corresponding location according to the semantic information. The results also show that all the images contain buildings.

We then evaluate the building contour detection using our method. The results show that in 74% of the cases the building is detected and in 5% of the cases, a part of the building is found. The building's shape, matching with the 2D GIS buildings layer, returns the correct poses in 26% of the cases with some wrong additional solutions.

Finally, **Figure 19** chart shows that DGPE finds in 21% of the cases a location with no orientation information and in 26% of the cases a camera pose that shows the correct building. In total, DGPE finds the image location in 47% of the cases. We then use the database of the image uploaded to "Flickr". We found that the text detection module returns in 32% of the cases text revealing geographical information. In 18% of the cases, the text is partially detected and in another 18% of the cases, the text is detected with additional text that does not reveal any localization information. When using the semantic information, we can notice that in 39% of the cases the building present in the image is found, and in 25% of the cases the correct building is found with additional locations. This information, based on text detection and other information downloaded from social media is not used in the results of Figure 21. The results also show that all the images contain buildings according to the "Google Vision API".

We can also see that in both Figure 20 and Figure 21, the building is detected in 32% of the cases and is partially detected in 29% of the cases. We then notice that the results generated using the image and GIS matching are successful in 11% to 14% of the cases. We can also find wrong detections in 18% to 21% of the cases. The rest of the matching cases return no results.

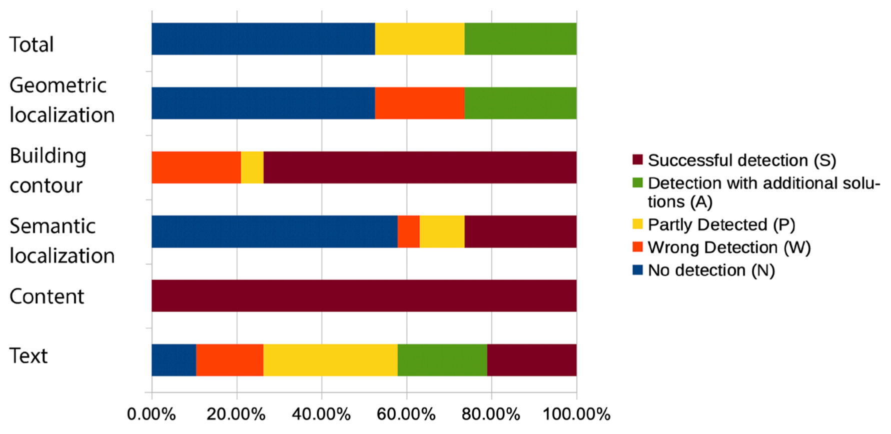


Figure 19. Chart summarizing DGPE results using the first image database.

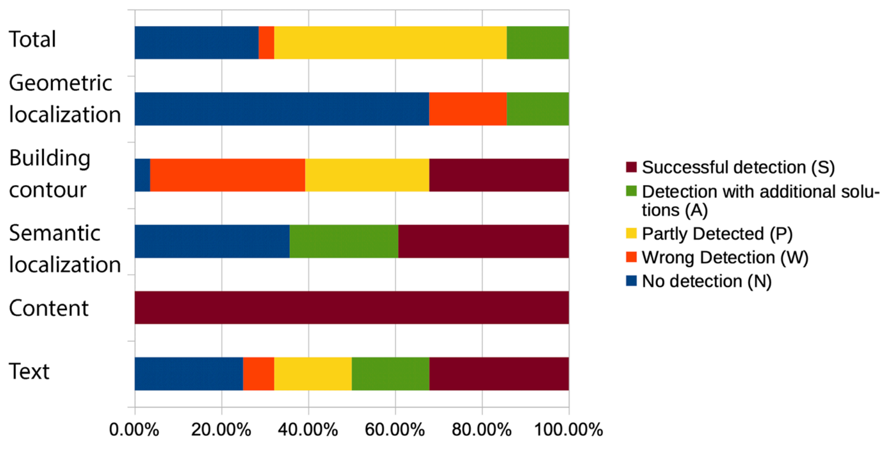


Figure 20. Chart summarizing DGPE results using the second image database with its associated semantic data.

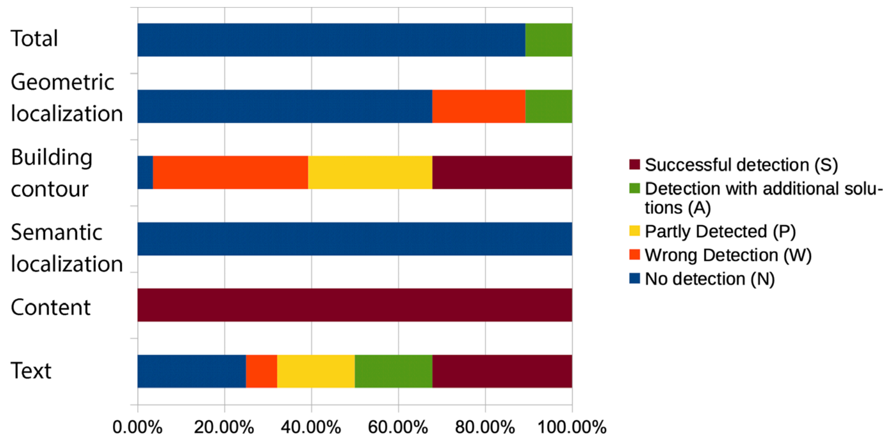


Figure 21. Chart summarizing DGPE results using the second image database with no associated semantic data.

The total DGPE results show that when using the same images database, we can only find the correct building in 11% of the cases without semantic information localization. On the other hand, when using semantic information from a detected text and the downloaded associated data, we can find in 14% a pose closer to the reference pose with an orientation pointing to the correct building. We can also find in 54% of the cases the localization information with no orientation information. Thus, we can improve the results of localization from 11%, when only using building and GIS matching, to 59% when using semantic information and matching the building shape to the GIS building layer.

5. Conclusion

In this paper, we have proposed the Data Gathering system for image Pose Estimation (DGPE) used to refine image localization and camera orientation estimation. We have used simple input data for DGPE because we believe this data is widely available and constantly updated on active sources, such as social media and collaborative cartography. We have explained our new building detection method SBBD and proven its robustness against some occlusion problems and weather changes. Finally, we have presented two case studies that show fully automatic results DGPE gave for image pose recognition. We have also presented the total DGPE results using two images databases. For the first database, only images are used as input data. Therefore, only 26% of the images' poses are found using the building shape and 2D GIS matching process. The text detected in the images is also useful in a few cases but needs preprocessing and filtering to be more relevant. For the second database, the images were uploaded on "Flickr", semantic data were added, such as image titles and hash tags. We compared with this second database the difference in localization when using semantic information or not. We have found that adding semantic information to the process improves the localization process by 48%, yet it does not add orientation information that could be found using the building shape and GIS matching.

The building was identified in only 11% of the cases when no semantic information was available. The same images' results were improved using the semantic data and 59% of the buildings were identified. We are looking forward to adding more functionality to DGPE by integrating logos and known landmarks detector. Using methods such as the ones described in [40] and [41] can improve our semantic understanding of the scene. Such algorithms help us choose the right area of the image in order to build a specific information query among the other modules. We are also thinking about improving our building detection method to recognize tilted roofs arcs and add the ability to detect multiple buildings. Finally, detected building geometries may also be compared with 3D GIS data. This may refine the pose accuracy by comparing the detected building facades to the 3D GIS data.

Acknowledgements

We would like to thank the Centrale Nantes School, the AAU-CRENAU Laboratory and the Lebanese University for supporting this project.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Grindgis (2015) 67 Important GIS Applications and Uses. http://grindgis.com/blog/gis-applications-uses

[2] The City of New York. NYC Open Data, 2017. https://data.cityofnewyork.us/City-Government/gis/x8zf-jmep

[3] INSPIRE. INSPIRE European Directive, 2017. http://inspire.ec.europa.eu

[4] Lyon City. Lyon Open Data Portal, 2018. https://data.grandlyon.com

[5] Auffray, C. (2015) Infographie—Portrait de l'utilisateurde smartphone français. http://www.zdnet.fr/actualites/infographie-portrait-de-l-utilisateur-de-smartphone-francais-39796286.htm

[6] Pew Research Center (2017) Mobile Fact Sheet. https://www.pewresearch.org/internet/fact-sheet/mobile/

[7] NainaKhedekar. We Now Upload and Share over 1.8 Billion Photos Each Day: Meeker Internet Report, 2014. https://www.firstpost.com/tech/news-analysis/now-upload-share-1-8-billion-photos-everyday-meeker-report-3652169.html

[8] Goodchild, M.F. (2007) Citizens as Sensors: The World of Volunteered Geography. *GeoJournal*, **69**, 211-221. https://doi.org/10.1007/s10708-007-9111-y

[9] Google Inc. (2017) Local Guides. https://maps.google.com/localguides/home

[10] Wing, M.G., Eklund, A. and Kellogg, L.D. (2005) Consumer-Grade Global Positioning System (GPS) Accuracy and Reliability. *Journal of Forestry*, **103**, 169-173. https://doi.org/10.1093/jof/103.4.169

[11] Chen, X.L., Shrivastava, A. and Gupta, A. (2013) NEIL: Extracting Visual Knowledge from Web Data. 2013 *IEEE International Conference on Computer Vision*

(*ICCV*), Sydney, 1-8 December 2013, 1409-1416. https://doi.org/10.1109/ICCV.2013.178

[12] Weyand, T., Kostrikov, I. and Philbin, J. (2016) Planet-Photo Geolocation with Convolutional Neural Networks. In: *European Conference on Computer Vision*, Springer, Berlin, 37-55. https://doi.org/10.1007/978-3-319-46484-8_3

[13] Suleiman, W., Favier, E. and Joliveau, T. (2011) Buildings Recognition and Camera Localization Using Image Texture Description. *International Journal of Computer Vision*, **61**, 159-184.

[14] Lowe, D.G. (2004) Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, **60**, 91-110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[15] Walch, F. (2016) Deep Learning for Image-Based Localization.

[16] Bioret, N., Servières, M. and Moreau, G. (2008) Outdoor Localization Based on Image/GIS Correspondence Using a Simple 2d Building Layer. 2*nd International Workshop on Mobile Geospatial Augmented Reality*, *LNGC*, Québec, 28-29 August 2008.

[17] Chu, H., Gallagher, A. and Chen, T. (2014) GPS Refinement and Camera Orientation Estimation from a Single Image and a 2D Map. 2014 *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Columbus, 23-28 Jun 2014, 171-178. https://doi.org/10.1109/CVPRW.2014.31

[18] Hochreiter, S. and Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation*, **9**, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[19] Taketomi, T., Sato, T. and Yokoya, N. (2011) Real-Time and Accurate Extrinsic Camera Parameter Estimation Using Feature Landmark Database for Augmented Reality. *Computers*: *Graphics*, **35**, 768-777. https://doi.org/10.1016/j.cag.2011.04.007

[20] Ventura, J., Arth, C., Reitmayr, G. and Schmalstieg, D. (2014) Global Localization from Monocular SLAM on a Mobile Phone. *Visualization and Computer Graphics*, *IEEE Transactions*, **20**, 531-539. https://doi.org/10.1109/TVCG.2014.27

[21] Zamir, A.R., Darino, A. and Shah, M. (2011) Street View Challenge: Identification of Commercial Entities in Street View Imagery. 10*th International Conference on Machine Learning and Applications and Workshops* (*ICMLA*), Volume 2, 380-383. https://doi.org/10.1109/ICMLA.2011.181

[22] Yelp. Yelp, 2017. http://www.yelp.com

[23] Yellowpages. Yellowpages, 2017. https://www.yellowpages.com/

[24] Gallup, D., Frahm, J.-M. and Pollefeys, M. (2010) Piecewise Planar and Non-Planar Stereo for Urban Scene Reconstruction. 2010 *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, 13-18 June 2010, 1418-1425. https://doi.org/10.1109/CVPR.2010.5539804

[25] Lafarge, F., Keriven, R., Bredif, M. and Hiep Vu, H. (2013) A Hybrid Multiview Stereo Algorithm for Modeling Urban Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**, 5-17. https://doi.org/10.1109/TPAMI.2012.84

[26] Hane, C., Zach, C., Cohen, A., Angst, R. and Pollefeys, M. (2013) Joint 3D Scene Reconstruction and Class Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Portland, 23-28 June 2013, 97-104.

[27] Hoiem, D., Efros, A.A. and Hebert, M. (2005) Geometric Context from a Single Image. *Tenth IEEE International Conference on Computer Vision*, Vol. 1, 654-661. https://doi.org/10.1109/ICCV.2005.107

[28] Hoiem, D., Efros, A.A. and Hebert, M. (2005) Automatic Photo Pop-Up. *ACM Transactions on Graphics*, **24**, 577. https://doi.org/10.1145/1073204.1073232

[29] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P. and Susstrunk, S. (2012) SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**, 2274-2281. https://doi.org/10.1109/TPAMI.2012.120

[30] Liu, B.Y., Gould, S. and Koller, D. (2010) Single Image Depth Estimation from Predicted Semantic Labels. 2010 *IEEE Conference Computer Vision and Pattern Recognition* (*CVPR*), San Francisco, 13-18 June 2010, 1253-1260.

[31] Wang, G.H., Chen, X.J. and Chen, S. (2014) Cut-and-Fold: Automatic 3D Modeling from a Single Image. 2014 *IEEE International Conference Multimedia and Expo Workshops* (*ICMEW*), Chengdu, 14-18 July 2014, 1-6. https://doi.org/10.1109/ICMEW.2014.6890555

[32] OpenStreetMap. Nominatim. OpenStreetMap, 2017. https://nominatim.openstreetmap.org

[33] U.S. Government (2017) GPS Accuracy. http://www.gps.gov/systems/gps/performance/accuracy

[34] Chen, H.Z., Tsai, S.S., Schroth, G., Chen, D.M., Grzeszczuk, R. and Girod, B. (2011) Robust Text Detection in Natural Images with Edge-Enhanced Maximally Stable Extremal Regions. *Proceedings International Conference on Image Processing*, *ICIP*, Brussels, 11-14 September 2011, 2609-2612. https://doi.org/10.1109/ICIP.2011.6116200

[35] Neumann, L. and Matas, J. (2012) Real-Time Scene Text Localization and Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Providence, RI, 16-21 June 2012, 3538-3545.

[36] Gómez, L. and Karatzas, D. (2014) MSER-Based Real-Time Text Detection and Tracking. 22*nd International Conference on Pattern Recognition*, Stockholm, 24-28 August 2014, 3110-3115. https://doi.org/10.1109/ICPR.2014.536

[37] Von Gioi, R.G., Jakubowicz, J., Morel, J.-M. and Randall, G. (2012) LSD: A Line Segment Detector. *Image Processing on Line*, **2**, 35-55. https://doi.org/10.5201/ipol.2012.gjmr-lsd

[38] Rother, C. (2002) A New Approach to Vanishing Point Detection in Architectural Environments. *Image and Vision Computing*, **20**, 647-655. https://doi.org/10.1016/S0262-8856(02)00054-9

[39] Zamir, A.R. and Shah, M. (2010) Accurate Image Localization Based on Google Maps Street View. In: *European Conference on Computer Vision*, Springer, Berlin, 255-268. https://doi.org/10.1007/978-3-642-15561-1_19

[40] Kirillov, A., Girshick, R., He, K.M. and Dollár, P. (2019) Panoptic Feature Pyramid Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, 15-20 June 2019, 6399-6408. https://doi.org/10.1109/CVPR.2019.00656

[41] Huang, Z.J., Huang, L.C., Gong, Y.C., Huang, C. and Wang, X.G. (2019) Mask Scoring R-CNN. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, 15-20 June 2019, 6409-6418. https://doi.org/10.1109/CVPR.2019.00657

The Following Abbreviations Are Used in This Manuscript

API:	Application programming interface
AR:	Augmented reality
DGPE:	Data gathering system for image pose estimation
EXIF:	Exchangeable image file format
GIS:	Geographical information systems
GNSS:	Global navigation satellite system
GPS:	Global positioning system
LOH:	Location orientation hypotheses
LSD:	Line segment detector
LSTM:	Long-short term memory
SBBD:	Segments Based Building Detection
SIFT:	Scale-invariant feature transform
SLAM:	Simultaneous localization and mapping
TICEP:	Tilt-Invariant Corner Edge Position
VGI:	Volunteered geographic information