Accurate Location of Façades of Interest in Street View Panoramic Sequences

André A. Araujo, Jonas C. Sampaio, Raphael S. Evangelista, Altobelli B. Mantuan, Leandro A. F. Fernandes
Instituto de Computação, Universidade Federal Fluminense (UFF)
CEP 24210-240, Niterói, RJ, Brazil
Email: {andrealvarado,revangelista}@id.uff.br, {jsampaio,amantuan,laffernandes}@ic.uff.br

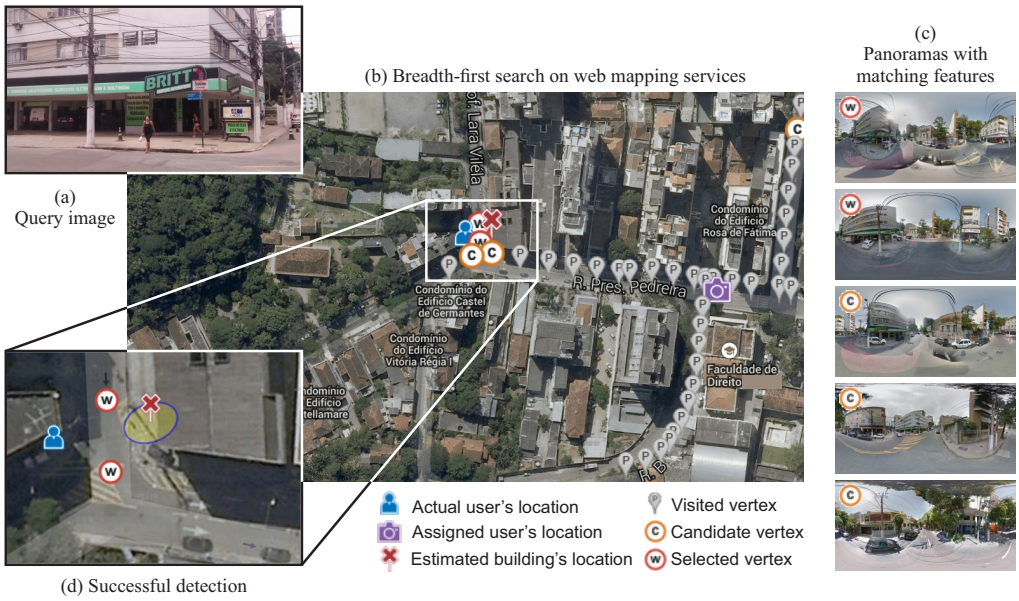

Fig. 1. Our system uses the graph of streets (b) and panoramic sequences (c) provided by Google Maps and Google Street View to estimate the location of a building of interest (d), identified by a query image (a). The search starts at inaccurate geographical coordinates associated to the given picture.

***Abstract*—Geo-spatial queries, i.e., queries that combine location data with other kinds of input, have taken huge importance in the last generation of search engines. The success of a geo-spatial search depends on the quality of the positioning information provided, for instance, by GPS-enabled smartphones. Therefore, the quality of the GPS signal and the quality of the built-in GPS may affect the accuracy of the estimated location, and hence the quality of the searching result. This paper proposes an automatic image-based solution for improving the estimation of the geographical coordinates of a building of interest on which a geo-spatial search will be performed. Our approach uses the inaccurate GPS coordinates estimated by smartphones as starting point for automated visual search into a graph of streets enhanced with street view panoramic sequences. During the search, our approach uses a query image of the building of interest to identify which panoramic views include the building's façade. From the geographical location of the panoramic views and from the best matching directions of the given image with the panoramic images, our approach triangulates the location of the target building. In addition, our approach estimates the uncertainty in the computed locations by modeling the error propagation along the triangulation procedure. We evaluate our method on several real images of buildings.**

***Keywords*-street view; geographical coordinates; environment map; feature extraction; feature matching; error propagation**

I. Introduction

The last years have seen a surge in the number of GPS-enabled devices launched on the market. Maybe because smartphones are digital Swiss Army knives that do just about everything. The use of this kind of device to search on the Internet is perhaps one of the most interesting alternatives to overcome, for instance, lack of knowledge on a particular subject and language issues while travelling abroad. However, the quality of geo-spatial search may be affected when the GPS has reduced accuracy. In urban centers, the GPS signal is usually of low quality when the device has the view of the sky partially blocked by canyons of buildings.

We consider the problem of inferring the location of a building of interest based on a picture of its façade, the inaccurate geographical coordinates provided by the GPS-camera, and street view panoramic sequences. The geo-spatial search considered in this work requires approximately correct coordinates of the building of interest to retrieve information about commercial establishments, restaurants, monuments, and cultural spaces in the target place.

Using a GPS-enabled smartphone, the user of our frame-

work takes a picture of a building (Fig.1a) with a geographical location estimative. Starting from the given location (in Fig.1b), our approach traverses the graph of streets provided by Google Maps, and compares the query image with environment maps provided by Google Street View. The visited vertices of the graph are identified by the , , and markers in Fig. 1. Vertices associated to environment maps having matching features with the given picture are denoted by and (Fig. 1c). The latter indicates vertices having the most important views of the façade of interest. We use the location of those vertices and the direction of the most likely correspondences between the query image and the environment maps to estimate the building's location ( ). The uncertainty on estimated coordinates are depicted by the yellow ellipse in Fig. 1d. Notice the distance between the actual location from where the picture was taken ( ), and the location assigned by the smartphone to the image file ( ). The inaccuracy intrinsic to off-the-shelf GPS-enabled mobile devices motivated the development of the proposed approach. The estimated location of the building can then be used as input in geo-spatial queries.

Contributions: This paper proposes a new image-based query procedure for estimating the geographical location of places of interest that include, but are not limited to, shops, monuments, and museums, from a query image and inaccurate GPS coordinates. The resulting location is expressed as a bivariate normal random variable that models the uncertainty in the computed geographical coordinates.

The main contributions of this paper are: (i) an algorithm for computing the location of buildings in a completely automatic way (Section II); (ii) a strategy for producing gnomonic projections of the environment that restrict the visual analysis of the panoramas to views of the sidewalks (Section II-A); (iii) a heuristic to select which are the environment maps in street view panoramic sequences that best estimate the location of the building of interest (Section II-C); and (iv) a derivation of how to estimate the error associated with the computed location (Section II-E).

A. Related work

Wikitude [1], Google+ Local [2], and MobiSpatial [3] are examples of frameworks that use the location provided by GPS-enabled smartphones, the orientation estimated by accelerometers, and digital compass to infer the relative location of nearby places registered in some mapping system. Wikitude [1] is an augmented reality API to present the approximate location of commercial establishments as an overlay of videos captured by smartphones. Google+ Local [2] retrieves the nearby establishments as the answer of a query provided by the user. Unfortunately, this query cannot be made using an image. MobiSpatial [3] benefits from location and orientation aware smartphones and existing open source spatial data initiatives to calculate a mobile user's visibility shape at his/her current location. This shape is used as query window to facilitate user interaction with the geo-spatial query process.

Micusik and Kosecka [4] presented a framework for creating 3D city models from street view panoramic sequences. The authors shown that by exploring image segmentation cues as well as presence of dominant scene orientations and piecewise planar structures, recurrent problems related to geometrical reconstruction of textureless surfaces may be overcome. In their work, the authors were not concerned to identify the places.

The problem of multiple view semantic segmentation for street view images was studied by Xiao and Quan [5]. This problem has many potential applications, such as to automatic vehicles and city modeling. Their approach uses structure from motion to reconstruct the scene geometry and prune incorrect correspondences. Xiao and Quan used a Markov random field to enable labeling of many images at the same time using the available geometry and color information.

Zamir and Shah [6] addressed the problem of finding the GPS location of images with an accuracy which is comparable to hand-held GPS devices. In their approach, the authors used only the panoramas provided by Google Street View and a query image to infer the actual location of the photographer.

To the best of our knowledge, Sampaio et al. [7], [8] were the first to propose an image-based framework to locate buildings using a query image, inaccurate GPS coordinates, and panoramic views provided by a web mapping service. The technique produces interesting results. However, the heuristics for panoramic view analysis, and for selecting the most important views of the target place produce unstable results.

B. Technique overview

Our technique aims at locating a building of interest by visually searching for its façade while traversing the graph of streets and environment maps provided by web mapping systems. We use breadth-first search (BFS) to traverse the graph, generate images of the streets by environment projection, and apply visual feature extraction and matching to compare the query image provided by the user with the generated projections of the environment. After detecting the vertices of the graph with best views of the target façade, the geographical coordinates of the building of interest are estimated by triangulating its location from known points and directions extracted from graph's data. Our approach is tailored to work with Goggle Street Map and Goggle Street View. However, it could be adapted to work with similar web mapping solutions. The execution of the whole process is schematized in Fig. 1. Details are presented in Section II. Implementation and results are discussed, respectively, in Sections III and IV.

In contrast to Sampaio et al. [8], our projections of the environment preserve the imaged straight lines. As a result, feature extraction and matching are not affected by image distortions. In addition, our approach triangulates the final location using two or more views of the place. Sampaio's et al. work is tailored to triangulate from two views. Also, we present an analytical derivation of uncertainty propagated along the computation chain that allows estimation of the error in the computed locations.

II. Proposed Approach

Let $G = \{V, E\}$ be a graph on a set of vertices V and a set of edges E. Each vertex in V is augmented with longitude

and latitude, such that it can be located in the geographic coordinate system. In addition, each vertex has assigned to it an equirectangular panoramic projection [9] of the environment, and an angle α_N that orients the panorama with respect to the North direction. Without loss of generality, we assume that edges in E represent sections of public pathways.

Our technique requires only two inputs provided by the user: a picture of the façade of interest (i.e, the *query image*), and the GPS coordinates from where the picture was taken. This data may be acquired at the same time by GPS-enabled smartphones. Usually, the device includes the coordinates in the Exif tag of the JPEG image file [10]. It is expected an image having the façade of interest as the main target. Due to restrictions imposed by the panoramas used in this project, it is expected a picture taken under daylight conditions.

The given GPS coordinates are the first guess we have to infer the actual location of the building of interest. In conventional web applications [1]–[3], the given location is used to rank results of simple queries on nearby places. However, the provided coordinates may not be valid under three conditions: (i) the given location may not be accurate enough due to calibration issues or the quality of the GPS system; (ii) the records of the web mapping system may suffer from some systematic error that relates the location of the camera to another place; and (iii) the façade of interest may actually be distant from the subject, perhaps a few kilometers away, but with a direct view to the camera.

For the first two conditions, one may expect that the given location will be different from the actual location of the photographer. It is likely that the building of interest will be close to the given location without attending the same city block or street. In such case, a reasonable alternative is to traverse the graph G by using the BFS strategy, assuming the closest vertex to the given location as starting point. Since we have a picture of the façade of interest, the stopping criteria could be the perception of being in a vertex that includes the façade in its panoramic view. The location of the last visited vertex could then be used in more interesting web queries.

The BFS is the basis of our searching approach. One contribution of our technique is how to perform the visual analysis of the panoramic views assigned to each visited vertex in order to identify the vertices with good views of the given façade (see Sections II-A to II-C for details). However, the problem with condition (iii) is that BFS may not be sufficient to guarantee that one will find a vertex of G that is close to the building of interest. Large distances would lead to very wide BFS to reach an interesting vertex. In order to avoid the drawback of distant pictures and to improve results on pictures taken at close range, we collect the subset of visited vertices with good views of the imaged façade while traversing the graph (see Sections II-B and II-C for details). From the location of those vertices and the direction that best match the query image, we triangulate the location of the building of interest (see Section II-D for details). The heuristic used for suppressing falsely collected vertices is a contribution of this work. Our approach also estimates the uncertainty in the computed locations by modeling the error propagation along the triangulation procedure (Section II-E).

The stopping criteria for our BFS approach are: (i) the detection of k vertices of G with a view to the intended façade; or (ii) the next vertex to be visited be located more than r_{max} meters from the starting vertex. In our experiments, we set $k = 5$ and $r_{max} = 200$ meters.

The distance (in meters) between two geographical locations $\mathbf{p}_1 = (x_1, y_1)$ and $\mathbf{p}_2 = (x_2, y_2)$ is computed using:

$$dist\,(x_1, y_1, x_2, y_2) = 12{,}742{,}018\,\tan^{-1}\left(\sqrt{\frac{a}{1-a}}\right), \quad (1)$$

where 12,742,018 is approximately twice the Earth's mean radius (in meters), and

$$a = \sin\left((y_2 - y_1)\,\frac{\pi}{360}\right)^2 + \cos\left(y_1\,\frac{\pi}{180}\right)\,\cos\left(y_2\,\frac{\pi}{180}\right)\,\sin\left((x_2 - x_1)\,\frac{\pi}{360}\right)^2.$$

In (1), x_i and y_i denote the longitude and latitude of the i-th input location (in degrees). It is important to emphasize that $\tan^{-1}(y/x)$ must be evaluated using the `atan2` function available in many programming languages, whose range is $[-\pi, \pi)$. It applies to all $\tan^{-1}(y/x)$ in this paper.

A. Computing views of the sidewalks from panoramas

We compute two images parallel to the sidewalks of the street (namely, the *left* and the *right views*, taken with respect to the front of Google's camera car) through gnomonic projections [9] of the environment assigned to a given map vertex. By construction, Google Street View registers the front view of the car at the center of the provided equirectangular panoramic images, i.e., $\lambda \approx 0$. The left, right and back views are registered at, respectively, $\lambda \approx -\pi/2$, $\lambda \approx \pi/2$, and $\lambda \approx \pm\pi$. As presented in Section II-B, the computed images are compared with the query image. From this comparison, we infer whether the façade of interest can be seen from a vertex of the graph. In our notation, $-\pi \leq \lambda < \pi$ and $-\pi/2 \leq \phi < \pi/2$ are, respectively, the coordinates for horizontal (Azimuth) and vertical (Zenith) fields of view of an equirectangular panorama with $W_{eq} \times H_{eq}$ pixels.

In order to prevent the registration of useless information of parts of the front, back, top, and bottom of the environment, we restrict the gnomonic projections to specific ranges of λ and ϕ. For the left view we assume $-\pi/2 - \Lambda \leq \lambda < -\pi/2 + \Lambda$, and for the right view we use $\pi/2 - \Lambda \leq \lambda < \pi/2 + \Lambda$. In both cases, $-\pi/2 - \Phi \leq \phi < \pi/2 + \Phi$, where Λ and Φ where experimentally set to $\Lambda = \pi\,5/18$ and $\Phi = \pi/6$.

Fig. 2 (top) shows the original equirectangular panorama assigned to one of the vertices marked with Ⓦ in Fig. 1b. The left and right gnomonic projections of the sidewalks produced for this panorama are presented in Fig. 2 (center). By comparing the gnomonic projections with the cubic face-concatenated panorama used in Sampaio et al. [8] (Fig. 2, bottom), one can see that our strategy preserves straight lines. The concatenation of the four lateral faces of a cubic environment, on the other

hand, map straight lines to C^0 continuity functions with joint points at faces' boundaries (see [8] for details).

We compute the intensities of the images of the sidewalks by casting rays from the pixels of the resulting images (sitting in planes tangent to a unit sphere) toward the center of the sphere, and by sampling the resulting position in the equirectangular panoramas with bicubic interpolation. For each pixel (i_{gn}, j_{gn}) in a given gnomonic image, where $i_{gn} \in \{0, 1, \cdots, W_{gn} - 1\}$ and $j_{gn} \in \{0, 1, \cdots, H_{gn} - 1\}$, the sampled coordinates $(x_{eq}, y_{eq}) \in \mathbb{R}^2$ in the equirectangular image are given by:

$$x_{eq} = \lambda_{eq} \frac{W_{eq} - 1}{2\,\pi} + \frac{W_{eq}}{2}, \text{ and} \tag{2a}$$

$$y_{eq} = \phi_{eq} \frac{H_{eq} - 1}{\pi} + \frac{H_{eq}}{2}, \tag{2b}$$

where $\lambda_{eq} = \tan^{-1}(w_{sh}/u_{sh})$, and $\phi_{eq} = \sin^{-1}(v_{sh})$. The coordinates of vector (u_{sh}, v_{sh}, w_{sh}) for the left and right images are computed, respectively, as $\left(u, \frac{v}{\sqrt{u^2+v^2+1}}, -1\right)$ and $\left(-u, \frac{v}{\sqrt{u^2+v^2+1}}, 1\right)$, where

$$u = \frac{2\,\tan(\Lambda)\;i_{gn}}{W_{gn} - 1} - \tan(\Lambda), \text{ and} \tag{3a}$$

$$v = \frac{2\,\tan(\Phi)\;j_{gn}}{H_{gn} - 1} - \tan(\Phi). \tag{3b}$$

In our experiments, we set the resolution of the resulting $W_{gn} \times H_{gn}$ images to:

$$W_{gn} = \left\lceil 2\,\frac{\tan(\Lambda)}{\tan\left(\frac{2\,\pi}{W_{eq}}\right)} \right\rceil \text{ and } H_{gn} = \left\lceil W_{gn}\,\frac{\tan(\Phi)}{\tan(\Lambda)} \right\rceil. \tag{4}$$

By doing so, we guarantee that the amount of aliasing introduced by the interpolation procedure will not affect the feature detection performed in the next step of the procedure. It is because the center of the resulting images keeps almost the same sampling rate of the original equirectangular panorama.

B. Detecting the target building in panoramic views

We use Affine-SIFT features [11] to compare the query image with each gnomonic projection of the sidewalks. According to our experience, traditional techniques like SIFT [12] and SURF [13] do not fit to our purposes since they are tailored to describe features invariant to translation, rotation, and scaling. Affine-SIFT, on the other hand, describes visual features invariant to affine transformations.

Fig. 2 illustrates the effects of the projection strategy adopted in this work and in [8] for building and analysing panoramic views. In this example, 190 matching features were detected in the query image and in the gnomonic projection of the left sidewalk (the white crosses in Fig. 2, center), against 18 matching features detected in the face-concatenated panorama (Fig. 2, bottom). As a result, the proposed strategy has a better chance of correctly identifying the building of interest in this environment map.

Fig. 2. Different projections of the environment map assigned to one of the preferred vertices in Fig. 1. From top to bottom: original equirectangular projection, left and right gnomonic projections produced by our approach, and cubic face-concatenated projection adopted in [8]. White crosses show the location of matching features. Image proportions where preserved.

In this work, a vertex of the graph G of streets is considered as facing the building of interest if at least one matching feature is detected in both the query image and in one of the two gnomonic projections of the sidewalks. Such a conservative heuristic may lead to the detection of spurious viewing vertices. Section II-C describes how false-viewing vertices are suppressed before the triangulation procedure (Section II-D).

For the triangulation procedure, one has to estimate the most likely direction of correspondence between the query image and a given panorama. Such a direction is characterized by the Azimuth λ_{high} (in the equirectangular map) having the highest concentration of Affine-SIFT features. We estimate λ_{high} as the global maximum of the circular histogram of abscissa of features mapped from the gnomonic projected images (i_{gn} in (3a)) to the equirectangular map (λ_{eq} in (2a)). In order to reduce noise, we take the maximum features count (w) after perform histogram smoothing with a Gaussian filter. In our experiments, we used a histogram with 360 bins and a filtering kernel window of size 9.

The preferential angle θ_{high} is computed by mapping λ_{high} from the equirectangular domain to the geographic domain with respect to the North direction (α_N) associated to each vertex of the graph. All angles in (5) are expressed in radians:

$$\theta_{high} = \lambda_{high} + \alpha_N. \tag{5}$$

C. Selecting the best panoramic views

Given the set C comprised of n vertices of G potentially facing the building of interest (i.e., the *candidate vertices*), one has to identify the subset $W \subset C$ with m vertices which are more likely to produce a good triangulation of the target façade (i.e., the *winner vertices*). We choose W as the subset of C that includes at least two vertices (condition 1) that maximize the mean preferred direction's weight (condition 2)

while having intra-set distances smaller than the threshold value d_{max} (condition 3). Recall that the preferred direction's weight w is computed for each vertex using the histogram of the procedure described in Section II-B.

Fig. 3 presents our simple and effective algorithm for selecting W, were $\mathcal{P}(C)$ denotes the power set of C, $|T|$ denotes the cardinality of T, and $\mathcal{D}(T)$ is a function that returns the greatest distance between pairs of vertices in T.

In our experiments, we assumed $d_{max} = 30$ meters. We choose this particular value in order to avoid the selection of subsets having vertices from different city blocks. When the resulting W is and empty set, our implementation skips the triangulation procedure and returns the coordinates of the vertex in C with highest weight w as the suggested closest location to the building.

Require: the set of candidate vertices C, and the threshold value d_{max}
$W \leftarrow \emptyset$
$\bar{w} \leftarrow 0$
for all $T \subset \mathcal{P}(C)$ **do**
  if $|T| \geq 2$ **and** $\mathcal{D}(T) \leq d_{max}$ **then**
    $\bar{t} \leftarrow$ mean preferred direction's weight of vertices in T
    if $\bar{w} < \bar{t}$ **then**
      $W \leftarrow T$
      $\bar{w} \leftarrow \bar{t}$
    end if
  end if
end for
return W

Fig. 3. Algorithm for suppressing spurious viewing vertices from the set C of vertices whose environments and query image include matching features.

D. Triangulating the target building

The location of the target building is computed as the point $\mathbf{t}$ that best fit the intersection of the straight lines $\mathbf{r}_i$, for $2 \leq i \leq m$, expanded by the location $\mathbf{p}_i = (x_i, y_i, 1)^T$ of each of the m winner vertices selected using the procedure described in Section II-C, and the direction $\mathbf{d}_i = (\sin(\theta_i), \cos(\theta_i), 0)^T$ indicating the most likely correspondence of the query image to the panorama assigned to the i-th vertex (see (5)).

It is important to recall that (x_i, y_i) and $(\sin(\theta_i), \cos(\theta_i))$ are expressed in the geographic coordinates system as angular values (i.e., longitude and latitude). However, since the working distances among $\mathbf{t}$ and $\mathbf{p}_i$ are up to few kilometers, one can assume that they are relatively small with respect to Earth radius ($\approx$ 6,371 kilometers). Without loss of generality, such observation allow us to treat $\mathbf{p}_i$ and $\mathbf{d}_i$ as vectors in a 2-dimensional homogeneous space. By doing so, the coefficients of the general equation of the line describing $\mathbf{r}_i$ are computed as:

$$\mathbf{r}_i = \begin{pmatrix} a_i \\ b_i \\ c_i \end{pmatrix} = \mathbf{p}_i \times \mathbf{d}_i = \begin{pmatrix} -\cos(\theta_i) \\ \sin(\theta_i) \\ \cos(\theta_i)\ x_i - \sin(\theta_i)\ y_i \end{pmatrix}, \quad (6)$$

where $\times$ denotes the cross product.

Denote by $\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_m)^T$ the $m \times 3$ matrix constructed from the coefficients of lines $\{\mathbf{r}_i\}_{i=1}^m$ (6). A point $\mathbf{t}$ that minimizes $\|\mathbf{R}\,\mathbf{t}\|^2$ is then the best fit to the data in an algebraic sense. Thus, the point $\mathbf{t}$ is obtained (up to scale) as the eigenvector of $\mathbf{R}$ with the smaller eigenvalue.

In order to obtain real valued eigenvalues and orthogonal eigenvectors, we evaluate the eigenvalues and eigenvalues of $\mathbf{R}$ by performing the singular value decomposition (SVD) on

$$\mathbf{S} = \mathbf{R}^T \mathbf{R}, \quad (7)$$

which is symmetric. $\mathbf{R}^T$ denote the transpose of $\mathbf{R}$. The singular vectors are then simply the eigenvectors and the square root of the singular values gives the eigenvalues of $\mathbf{R}$.

Denote by $\mathbf{v} = (x_\mathbf{v}, y_\mathbf{v}, z_\mathbf{v})^T$ the eigenvector of $\mathbf{R}$ (and $\mathbf{S}$) with the smaller eigenvalue. The location $\mathbf{t}$ of the building of interest is computed by normalizing the homogeneous coordinate of $\mathbf{v}$ to 1:

$$\mathbf{t} = \begin{pmatrix} x_\mathbf{t} \\ y_\mathbf{t} \end{pmatrix} = \begin{pmatrix} x_\mathbf{v}/z_\mathbf{v} \\ y_\mathbf{v}/z_\mathbf{v} \end{pmatrix}. \quad (8)$$

E. Error propagation

Any given computation propagates the uncertainties associated with its input variables to its output. The uncertainty associated with a location $\mathbf{t}$ whose coordinates are computed from a set of experimental data $\vartheta = \{x_1, y_1, \theta_1, x_2, y_2, \theta_2, \cdots, x_m, y_m, \theta_m\}$ can be approximated by first-order error propagation model [14]:

$$\Lambda_\mathbf{t} = \nabla_\mathbf{t}\, \Lambda_\vartheta\, \nabla_\mathbf{t}^T, \quad (9)$$

where $\Lambda_\mathbf{t}$ is the covariance matrix that models the uncertainty in $\mathbf{t}$, $\nabla_\mathbf{t}$ is the Jacobian matrix for the function that computes each term of $\mathbf{t}$ from the $3\,m$ input variables in ϑ, and Λ_ϑ is the covariance matrix that models the uncertainty in the input variables. Using such a model, we can obtain elliptical confidence intervals for the computed location $\mathbf{t}$, which correspond to the k-sigma band around the mean ($\mathbf{t}$) in a bivariate normal distribution with covariance matrix $\Lambda_\mathbf{t}$.

To apply the error propagation model (9), one needs to estimate the uncertainty associated with each input variable and to compute the Jacobian matrix for the equation that calculates the geographical coordinates of $\mathbf{t}$.

In our approach, we assume that the uncertainties on the location $\mathbf{p}_i$ of each of the m interesting vertices retrieved from Google Street View are independent bivariate normal distributions with standard deviations σ_{x_i} and σ_{y_i} in, respectively, longitude and latitude coordinates. Likewise, we assume independent identically distributed uncertainty in the θ_i angles, i.e., $\sigma_{\theta_i} = \sigma_\theta$. As a result, Λ_ϑ is a diagonal $3\,m \times 3\,m$ matrix:

$$\Lambda_\vartheta = diag\left(\sigma_{x_1}^2, \sigma_{y_1}^2, \sigma_\theta^2, \cdots, \sigma_{x_m}^2, \sigma_{y_m}^2, \sigma_\theta^2\right). \quad (10)$$

The Jacobian matrix of the function that computes $\mathbf{t}$ (8) is:

$$\nabla_\mathbf{t} = \begin{pmatrix} \partial_{x_1} x_\mathbf{t} & \partial_{y_1} x_\mathbf{t} & \partial_{\theta_1} x_\mathbf{t} & \cdots & \partial_{y_m} x_\mathbf{t} & \partial_{\theta_m} x_\mathbf{t} \\ \partial_{x_1} y_\mathbf{t} & \partial_{y_1} y_\mathbf{t} & \partial_{\theta_1} y_\mathbf{t} & \cdots & \partial_{y_m} y_\mathbf{t} & \partial_{\theta_m} y_\mathbf{t} \end{pmatrix}, \quad (11)$$

where the partial derivatives $\partial_\upsilon x_\mathbf{t}$ and $\partial_\upsilon y_\mathbf{t}$, for $\upsilon \in \vartheta$, can be solved using the chain rule.

According to Section II-D, the first stage of the triangulation process computes the coefficients of straight lines $\mathbf{r}_i$ (6). Their partial derivatives are:

$$\partial_\upsilon a_i = \begin{cases} \cos(\theta_i) & \text{for } \upsilon = x_i \\ 0 & \text{otherwise} \end{cases}, \tag{12a}$$

$$\partial_\upsilon b_i = \begin{cases} -\sin(\theta_i) & \text{for } \upsilon = y_i \\ 0 & \text{otherwise} \end{cases}, \text{ and} \tag{12b}$$

$$\partial_\upsilon c_i = \begin{cases} \sin(\theta_i) & \text{for } \upsilon = x_i \\ \cos(\theta_i) & \text{for } \upsilon = y_i \\ -y_i \cos(\theta_i) - x_i \sin(\theta_i) & \text{for } \upsilon = \theta_i. \\ 0 & \text{otherwise} \end{cases} \tag{12c}$$

The matrix $\mathbf{S}$ is computed from $\{\mathbf{r}_i\}_{i=1}^{m}$ using (7). The partial derivatives of $\mathbf{S}$ are calculated as:

$$\partial_\upsilon \mathbf{S} = \partial_\upsilon \mathbf{R}^T \mathbf{R} + \mathbf{R}^T \partial_\upsilon \mathbf{R}, \tag{13}$$

where $\partial_\upsilon \mathbf{M}$ denotes the partial derivatives of the coefficients of the matrix $\mathbf{M}$ with respect to a given variable $\upsilon \in \vartheta$. In (13), $\partial_\upsilon \mathbf{R}^T = (\partial_\upsilon \mathbf{R})^T$, and $\partial_\upsilon \mathbf{R}$ is computed just by placing the derivatives of $\mathbf{r}_i$ (12) in the respective lines of $\partial_\upsilon \mathbf{R}$.

The next stage of the triangulation procedure computes the eigenvectors and eigenvalues of $\mathbf{R}$ from the SVD of $\mathbf{S}$. We used the derivations presented by Giles [15] to compute the partial derivative matrix $\partial_\upsilon \mathbf{W}$ of the matrix of eigenvectors:

$$\partial_\upsilon \mathbf{W} = \mathbf{W} \left(\mathbf{F} \circ \left(\mathbf{W}^{-1} \partial_\upsilon \mathbf{S}\, \mathbf{W} \right) \right), \tag{14}$$

where $\mathbf{F}_{r,c} = (d_c - d_r)^{-1}$ for $r \neq c$, and zero otherwise, $\mathbf{W}^{-1}$ denotes the inverse of matrix $\mathbf{W}$ (its transpose, since $\mathbf{W}$ is orthogonal), $\circ$ denotes the Hadamard product of two matrices of the same size, defined by each element being the product of the corresponding elements of the input matrices, and $\{d_1, d_2, d_3\}$ are the eigenvalues of $\mathbf{S}$.

Finally, the partial derivatives of $\mathbf{t}$ (11) are:

$$\partial_\upsilon x_\mathbf{t} = \frac{\partial_\upsilon x_\mathbf{v}}{z_\mathbf{v}} - \frac{x_\mathbf{v}\, \partial_\upsilon z_\mathbf{v}}{z_\mathbf{v}^2}, \text{ and} \tag{15a}$$

$$\partial_\upsilon y_\mathbf{t} = \frac{\partial_\upsilon y_\mathbf{v}}{z_\mathbf{v}} - \frac{y_\mathbf{v}\, \partial_\upsilon z_\mathbf{v}}{z_\mathbf{v}^2}. \tag{15b}$$

Given the covariance matrix Λ_ϑ (10) and the Jacobian matrix $\nabla_\mathbf{t}$, we estimate the uncertainty in the location of the target building using (9). In this equation, Λ_ϑ stores the uncertainty in the input data, while $\nabla_\mathbf{t}$ weights the influence of these uncertainties in the computed geographical coordinates.

We used $\sigma_\theta = \pi/135$, $\sigma_{x_i} = 1/dist\,(x_i, y_i, x_i + 1, y_i)$, and $\sigma_{y_i} = 1/dist\,(x_i, y_i, x_i, y_i + 1)$ in all experiments, where $dist$ is defined by (1). By assuming local flatness on Earth's surface, one can verify that σ_{x_i} and σ_{y_i} were set in order to ensures an interval of ± 3 meters with 99.7% of confidence to the location of the i-th vertex considered. The confidence interval assumed for the preferential direction (θ) was ± 4 degrees due to the window size of the smoothing filter applied to the circular histogram of features' location (Section II-B).

Fig. 4. Thumbnail version of the query images used in the experiments. The collection includes 30 images of 27 buildings.

III. Implementation

We have implemented the proposed algorithms using Java. We used the ImageJ library [16] to perform bicubic interpolation of panoramas, and EJML [17] to work with matrices and to compute SVD. We used the reference C++ implementation of the Affine-SIFT provided by Morel and Yu [11]. Our system uses the Maps API [18] to access Google Maps and Google Street View, and to plot results.

The feature extraction is the bottleneck of our framework. In order to alleviate the computational cost of recurring searches on the same areas of the map, we have implemented a caching mechanism to store the features extracted from panoramic views accessed in previous executions of our system. For this purpose, we have modified the original implementation of Affine-SIFT to perform feature extraction and feature matching as separated procedures. Those procedures are called from our Java application by command-line prompt commands.

IV. Results and Discussion

We validate our technique by analyzing the results produced by processing 30 pictures of 27 places of our city. Fig. 4 shows thumbnail versions of the query images used in the experiments. The pictures were taken at different lighting conditions, with varying distances from the target, and with resolution of $2{,}592 \times 1{,}456$ pixels. The input data was acquired using a smartphone Motorola Moto G. The experiments were performed on a Linux 2.4 Ghz Intel Dual Core machine with 2 GB of RAM and Internet access.

TABLE I
DETECTION RESULTS FOR IMAGES IN FIG. 4. DISTANCE IN METERS. * AND † DENOTE, RESPECTIVELY, CASES WHERE ONLY ONE PANORAMIC VIEW INCLUDES THE TARGET BUILDING AND IMPOSSIBLE CASES.

Case	*dist* (📍,📷)	📍	©	Ⓦ	Success
1	133.89	45	3	2	Yes
2	17.80	17	1	2	Yes
3	14.55	26	1	2	Yes
4	10.76	19	0	2	Yes
5	4.71	21	3	2	Yes
6	9.54	38	2	2	Yes
7	196.52	152	3	2	Yes
8	39.34	26	3	2	Yes
9	48.27	25	3	2	Yes
10	14.55	10	0	2	Yes
11	50.86	38	3	2	Yes
12	34.37	30	1	2	Yes*
13	33.34	25	1	1	Yes*
14	35.00	41	1	1	Yes*
15	7.91	37	0	1	Yes*
16	20.37	30	1	1	Partial
17	263.64	14	1	1	Partial
18	399.16	188	4	1	Partial
19	296.23	40	0	1	Partial
20	15.67	22	0	1	No
21	3.86	22	0	0	No
22	48.97	32	0	1	No
23	31.84	12	3	2	No
24	27.23	24	0	2	No
25	265.87	159	4	1	No
26	646.46	25	1	1	No
27	25.69	7	3	2	No
28	9.47	103	3	1	No†
29	8.07	21	0	0	No†
30	269.75	42	1	1	No†

Table I summarizes the results for testing cases in Fig. 4. The order of the images correspond to the order of table's rows. In Table I, the second column shows the distance (in meters) between the actual location of the photographer and the location assigned to the query image. Columns 3 to 5 show, respectively, the number of visited vertices that had no matching features, the number of vertices with at least one matching feature and not used by the triangulation procedure, and the number of vertices used to compute the location of the target building. The last column shows the detection result.

For cases 1 to 15, the detection of target buildings was successfully accomplished. We classify as success the triangulation of the façade of interest whose uncertainty ellipse includes its correct location. Figs. 1, 5, and 6 illustrate results for cases 1, 6, and 9, respectively. In cases 12 to 15, Google Street View provides only one panoramic view for each target façade. As a result, our system was not capable to triangulate the target building. However, it was capable to identify this single vertex and return its location as the suggested closest location for geo-spatial queries. Refer to Fig. 7 for an example. Cases 12 to 15 are successful under the available data.

The detection results of cases 16 to 19 were partially successful. For case 16, Google Stret View provided more than one environment map including the target buildings. Our technique was capable to identify only one of them as a good match. Thus, the triangulation procedure was not performed, but our system improved the location estimated by GPS. In cases 17 to 19, the coordinates assigned to the query image are more than 200 meters away from the correct user's location. Fig. 8 illustrates that situation for case 18. By increasing the searching radius, our system was capable to locate the façades of interest. In Table I, columns 3 to 5 show the vertex count for the first run of the approach, i.e., using $r_{max} = 200$ meters.

Our approach failed in cases 20 to 27 due to feature matching issues. Cases 20 to 23 reported only incorrect candidate vertices. One correct and one incorrect vertex were used to triangulate in case 24. The GPS estimated the camera position more than 200 meters away from the correct user location in cases 25 and 26. Unfortunately, our system was not capable to identify the correct matching environments even after increasing the searching radius. Finally, in case 27 our system was capable to identify more than two correct candidate vertices for triangulation. However, the two winner vertices pointed to different parts of the building, leading to divergent preferential directions. As a result, the intersection of the lines defined by each vertex location and preferential direction happened on the opposite side of the street.

The last three lines in Table I represent cases that cannot be handled by our system. In case 28, the user took the picture of a tree. The façade in case 29 was not registered by Google Street View because there is a newsstand blocking the view of the only panorama that would record the target building. The query image in case 30 was taken at night.

A. Limitation

Google Street View provides panoramas captured under daylight conditions. In addition, Affine-SIFT is only partially tolerant to illumination changes. Therefore, our framework fails when the query image is taken at night. The last image in Fig. 4 (case 30 in Table I) illustrates this situation. Textureless façades also affect the detection capacity of our framework. It is because visual features have dependence upon texture variability. That led to divergent directions in testing case 27.

The response time to each query image vary depending on the speed of the Internet connection and the number of visited panoramas. The bottleneck of our implementation is the feature extraction procedure performed for each gnomonic projection of the panorama. Each run of the Affine-SIFT procedures takes approximately 64 seconds. The use of a caching mechanism helps to alleviate this issue.

Finally, when only one panoramic view of the target façade is available in the web mapping system, our approach is unable to triangulate the location of the target building.

V. CONCLUSION

In this paper, we presented a completely automatic approach that uses panoramas, a single query image, and inaccurate geographical coordinates of the camera for estimating the geographical location of buildings. We have introduced the use of gnomonic projection to analyze only the most important portion of street view panoramic sequences. The analysis was performed by comparing the Affine-SIFT features extracted from gnomonic projections of the environment with features

Fig. 5. Detection result for case 6. The target was successfully detected, as can be seen in the detailed view of the uncertain ellipse including its façade.

Fig. 6. Detection result for case 9. Notice that a candidate vertex was excluded from the triangulation procedure, even being close to the winner vertices. Our approach was capable to identify this vertex as a false match to the query image. Its preferred direction points to the beach.

Fig. 7. Detection result for case 13. Sparse candidate vertices were reported. Our approach retrieved the coordinates of the best matching panorama as the suggested closest location to the target building.

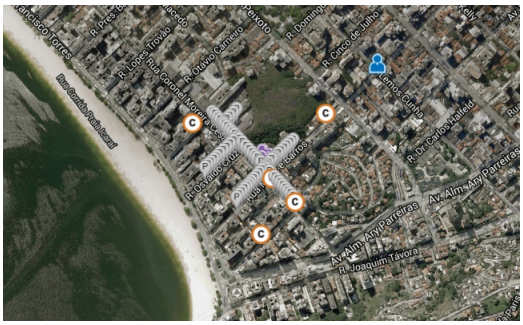

Fig. 8. Detection result for case 18. In this example, the actual user's location and the location assigned to the query image exceeded the searching radius threshold. By increasing r_{max} the target façade would be detected. Due to Maps API limitations, some path markers were removed from plot.

extracted from the query image. Also, we have developed an approach to indicate which panoramic views must be combined to agree on a good identification of the façade of interest, and have used SVD to triangulate the location that best fit in an algebraic sense the views of the façade. In addition, we presented an analytical derivation of uncertainty propagated along the computation chain that allows estimation of the error in the computed locations.

We demonstrated the effectiveness of our techniques by implementing the proposed framework and using it to locate several buildings in real case examples. The experimental validation have shown that the proposed approach is accurate and precise when more than one panoramic view of the building of interest is available in the mapping system.

Acknowledgment

The authors thank Kelly Bentes for her assistance. This work was partially sponsored by FAPERJ (E-26/110.092/2014, E-26/110.484/2014). A. A. Araujo is sponsored by a FAPERJ fellowship (E-26/100.991/2014). J. C. Sampaio is sponsored by a CNPq-Brazil fellowship. R. S. Evangelista and A. B. Mantuan are sponsored by CAPES. L. A. F. Fernandes has a CNPq-Brazil fellowship (308316/2014-2).

References

[1] Wikitude GmbH, "Wikitude App," http://www.wikitude.com/, 2015.
[2] Google Inc., "Google+ Local," https://plus.google.com/local/, 2014.
[3] J. Yin and J. Carswell, "MobiSpatial: open source for mobile spatial interaction," in *Proc. of the ACM SAC*, 2012, pp. 572–573.
[4] B. Micusik and J. Kosecka, "Piecewise planar city 3D modeling from street view panoramic sequences," in *Proc. of the IEEE CVPR*, 2009, pp. 2906–2912.
[5] J. Xiao and L. Quan, "Multiple view semantic segmentation for street view images," in *Proc. of the IEEE ICCV*, 2009, pp. 686–693.
[6] A. R. Zamir and M. Shah, "Accurate image localization based on Google Maps Street View," in *Proc. of the ECCV*, 2010, pp. 255–268.
[7] J. C. Sampaio, R. S. Evangelista, and L. A. F. Fernandes, "Locating façades using single perspective images, cube maps and inaccurate GPS coordinates," in *Workshop of Undergraduate Works of SIBGRAPI*, 2013.
[8] ——, "Determining the location of buildings given a single picture, environment maps and inaccurate GPS coordinates," in *Proc. of the ACM SAC*, 2015.
[9] D. H. Maling, *Coordinate Systems and Map Projections*, 2nd ed. Butterworth-Heinemann, 1993.
[10] JEITA, "Exchangeable image file format for digital still cameras: Exif Version 2.2," Tech. Rep. JEITA CP-3451, April 2002.
[11] G. Yu and J. M. Morel, "ASIFT: a new framework for fully affine invariant image comparison," *SIAM J. Imaging Sci.*, vol. 2, no. 2, 2009.
[12] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. of the IEEE ICCV*, 1999, pp. 1150–1157.
[13] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.
[14] L. G. Parratt, *Probability and Experimental Errors in Science*. John Wiley and Sons Inc., 1961.
[15] M. B. Giles, "An extended collection of matrix derivative results for forward and reverse mode automatic differentiation," Oxford University, Tech. Rep. 1079, 2008.
[16] W. Rasband, "ImageJ: Image Processing and Analysis in Java (version 1.48)," http://imagej.nih.gov/ij/, 2015.
[17] P. Abeles, "EJML: Efficient Java Matrix Library," http://ejml.org/, 2015.
[18] Google Inc., "Maps API," https://developers.google.com/maps/, 2015.