Hierarchical Semantic Hashing: Visual Localization from Buildings on Maps

Turgay Senlet, Tarek El-Gaaly, Ahmed Elgammal
Department of Computer Science
Rutgers University
Piscataway, USA
{tsenlet, tgaaly, elgammal}@cs.rutgers.edu

***Abstract*—In this paper we present a vision-based method for instant global localization from a given aerial image. The approach mimics how humans localize themselves on maps using spatial layout of *semantic* elements on the map. Unlike other matching and localization methods that use visual appearance or feature matching, our method relies on robust and consistently detectable semantic elements that are invariant to illumination, temporal variations and occlusions. We use the buildings on the map and on the given aerial query image as our semantic elements. Spatial relations between these elements are efficiently stored and queried under a hierarchical semantic version of the Geometric Hashing algorithm that is inherently rotation and scale invariant. We also present a method to obtain building locations from a given query image using image classification and processing techniques. Overall this approach provides fast and robust localization over large areas. We show our experimental results for localizing satellite image tiles from a 16.5 km sq dense city map with over 7,000 buildings.**

Keywords—visual localization; geometric hashing; satellite map; building detection; geographical information systems

I. INTRODUCTION

Visual localization is the problem to determine the position of a camera with respect to a global or local frame of reference, by using visual sensing of the environment. The presence of camera noise, illumination changes, perspective effects and temporal variations (*e.g.* seasonal changes) adversely affect the accuracy of visual localization thus making it a challenging problem, particularly in the case of localizing within large-scale images.

Given an orthographic aerial view captured by a ground-facing camera, if the image does not have geo-location information, it is difficult to recognize where this view lies in a large geographic area. This image can either be taken by Unmanned Aerial Vehicles (UAV), surveillance satellites, airplanes or simply downloaded from the Internet. The latter case is very common nowadays with the plethora of images on the Internet and the demand to understand this data.

A semantic-level understanding of maps better mimics the way humans localize on maps and it is more feasible when compared to conventional image matching techniques. Searching on large maps using local image features, cross-correlation or bag-of-words techniques are impractical. These algorithms may fail because of the sheer number of visual features and amount of similar regions, like similar rooftops, roads and buildings. Furthermore, matching between different data modalities (such as GIS (Geographic Information System) maps, satellite images, maps and aerial images) with these methods become a problem due to very different appearances of the same regions.

We claim that a key feature that discriminates regions in these environments is the structure or layout of semantic elements, such as buildings, roads, sidewalks, trees, etc. From set of possible semantic elements, because of their abundance on residential areas and existence of their ground truth locations, we chose to use buildings as our semantic features.

In this paper, we present an instantaneous visual localization method based on the semantic structures –namely building locations- found on maps. We use a form of geometric hashing that utilizes semantic information. The localization method is able to localize a single view within a large-scale map containing up to 7,000 buildings. The map covers a 6.5 x 2.5 km (16.5 km^2) geographic area, which is a portion of downtown Seattle, WA.

Our localization method consists of hashing and querying stages (Fig. 1). Hashing stage is the offline process, where the known building locations on the map are used to build an efficient hash table structure for fast and robust querying. In the hashing step, we store relative positions and properties of buildings with respect to their close neighbors. Querying stage is performed online and provides a single-shot global localization using an orthographic query image of a small area with buildings. Buildings on the query image are extracted by our proposed building detection method. Resulting buildings and their geometric layouts are used to obtain queries for the hash table from hashing stage. After the query, we receive votes for possible locations on the map and further checks can be done to determine the best matching location.

The contribution of this paper is two-folds. We built a novel hashing scheme and a full pipeline around it to enable instantaneous localization in large city environments and proposed a satellite image based building detection algorithm that uses image processing techniques. We used semantic features in images to perform fast and efficient localization on large-scale maps. The approach can be extended to work on arbitrary projective top view images and very well suited for visual localization systems on MAVs (micro aerial vehicles).

II. Related Work

The broad category of computer vision methods that provide global localization using only a single observation (*i.e.* instantaneous), can use feature matching, BoW (bag-of-words) [1], map-matching (template-matching/correlation) [2] and image retrieval algorithms [3]. Feature matching and BoW algorithms come up with efficient ways to search features coming from query observations in a vast, prebuilt feature set of the environment. BoW can be thought of as attempting to capture higher-level groups of features that commonly occur together in the environment and by this, it takes a step closer to semantic-level features. On large urban maps, a lot of visually similar regions occur. This is a challenge for feature matching algorithms. In Fig. 3 we can see the similarity in the majority of the map. Sliding window methods like template-matching, suffer in performance with increased scale. They become even in orders of magnitude more computationally expensive when searched over possible rotations and scales. They are also more susceptible to appearance changes like view angle, lighting and seasonal changes. There is a large body of work on image retrieval algorithms (a comprehensive list is given in [3]), where the class of content-based image retrieval (CBIR) techniques to find the closest looking image to a given image can be employed for localization purposes. Unfortunately many of the CBIR methods rely on the basis of the aforementioned methods and have similar shortcomings.

Semantic-level features are more robust against visual uncertainties in uncontrolled environments than local point features. Methods in [4] have focused on building semantic representations of indoor environments using the presence of informative objects. There are other approaches that focused on constructing semantic-level information from low-level features [5] using supervised learning in order to perform image retrieval.

Methods for localizing camera images using street view images and satellite maps are proposed in [6] and [7] respectively. For querying, [6] uses SIFT features of street view images stored on a tree structure and [7] uses a Bayesian tracking framework to iteratively localize vehicle position by matching camera view to satellite map, but does not attempt to solve the instantaneous or initial localization problem.

One of the most relevant work is the method proposed in [8] for visual localization on satellite imagery using low-level local SIFT features. This work uses feature matching and a BoW-type approach to quantize the features. Unfortunately, matching performance is only evaluated for visually discriminative areas of the map and approach does not address disambiguating between similar regions for localizing in urban environments. The search is will suffer in performance as search and query images grow. The approach is also affected from scale and rotation and has to search over multiple scales.

Map matching based UAV localization in a tracking framework is presented in [9], where image-to-image and image-to-map registration is performed iteratively. SLAM from downward facing camera has been affectively applied to the recently growing field of UAV localization [10]. These approaches are for rather small-scale environments and do not provide instantaneous global localization on maps.

Although Geometric hashing has not been directly used for general map based localization methods, it has been used as coarse global localization methods by several works in other scenarios: edge based indoor localization, where each room is a model [11] and building-facade based 2D localization on maps [12]. The latter work uses hand-made ground truth of building facades on satellite images and matches them to buildings seen in images taken from a ground robot. Although this method is promising in its nature, it does not scale to larger areas nicely. Experiments in this work are performed on a small number of buildings and accuracy of the method depends on the accuracy of detailed hand-made façade data.

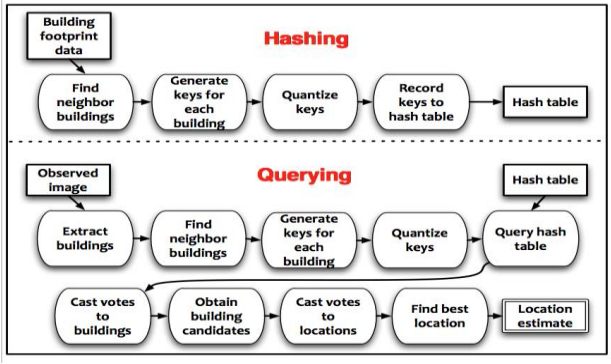


Fig. 1 Algorithm overview. Algorithm consists of two main stages; offline hashing and online image querying.

III. Semantic Geometric Hashing for Localization

In this chapter we first describe the construction of the semantic geometric hash table that encodes the layouts of small neighborhoods of buildings on the map. We then describe at query time, given an observation image, how we localize this image by extracting the buildings from the image and finding the closest location to the query image using the semantic geometric hash table.

A. Constructing the Semantic Hash Table

Geometric hashing [13] has been proposed as a model-based efficient object recognition scheme that has later been used for many pattern recognition tasks including 3D shape recognition [14], pose estimation, hand-written image matching; generally by using point or line features [15], [16].

Unlike the traditional geometric hashing applications that use low-level point or line features in an image, we present an algorithm that makes use of higher-level semantic features in the scene. We hash the building centroid locations and building features such as the area ratios. This approach is more robust against noise or detection errors when compared to using low-level visual features in the image. Furthermore using semantic features enables us to match across different data modalities, *e.g.* matching satellite images to GIS building contour polygon data or other street maps data.

A *model* in geometric hashing is the set of feature points of a single entity to be detected in the scene. Performance of hashing degrades when the feature points to be hashed in the model increases too much. Thus, we chose to have each model to correspond to a building on the map and its K nearest

neighbor buildings. For each building, we find at most K nearest neighbors inside a neighborhood R (Fig. 2) to be the model for that building. The hashing step is where we store the geometric inter-relations (*i.e.* layout/structure) of the elements of these models. In the query stage, when one of the models is detected, this will suggest localizing the center (key) building of that model. We build our hash table using ground truth locations and areas of buildings obtained from GIS data.

Detailed steps of hashing buildings are given in Algorithm 1. To normalize the coordinates of neighbor buildings in the model, we take one neighbor building to be the base neighbor to form a baseline with the key building (Fig. 2). Each neighbor becomes a base neighbor, where rest of the neighbors are normalized based on this baseline. Normalization is performed by considering a new coordinate system where the key building center is the origin (0,0) and base neighbor center is the point (1,0). Normalized coordinates of neighbor buildings along with the area ratios to key and base neighbor buildings are quantized to generate hashing keys for the model. Area ratios are used to generate more unique hash keys and like the normalized locations, this feature is also translation, scale and rotation invariant. Corresponding to each key, we note down key building number and base neighbor number in the hash table. The K nearest neighbor approach and the selected baseline with center point being fixed, reduces the per model geometric hashing complexity to $\mathbf{O}(K^2)$. In the original geometric hashing algorithm, for a model with K points, all possible point pairs in a model (all neighbor pairs) are used as basis, which leads to $\mathbf{O}(K^3)$ per model complexity. Where n is the number of buildings to be hashed in the whole map, complete hashing time complexity for our approach becomes $\mathbf{O}(nK^2)$.

Geometric hashing is inherently invariant to scale and rotation due to its normalized coordinate system and it is also robust against occlusion and missing data points since multiple entries for a model is stored in the hash table and using a voting scheme, model with the maximum number of votes is selected as the best model.

B. *Querying and Localization Stage*

1) Automatic Building Detection from Satellite Images

In order to be able to query the location of a given image, we need to extract the semantic features from the image for semantic hashing queries. By processing a given query image, we automatically determine building locations and sizes by employing a combination of machine learning and image processing techniques. Although the query images can come from a variety of different sources like UAV's or aerial imagery, for our testing purposes, because of their availability and ease of access, we used satellite images for queries as shown in Fig. 4 (a).

Automatic and semi-automatic building footprint detection from aerial imagery is a well-studied area in the Geographic Information Systems and Remote Sensing (GIS-RS) literature. Usually building detection in Remote Sensing is performed using the help of other sensors and geographic information like LIDAR, Digital Elevation Maps (DEM) and multi-spectral imagery [17]-[19].

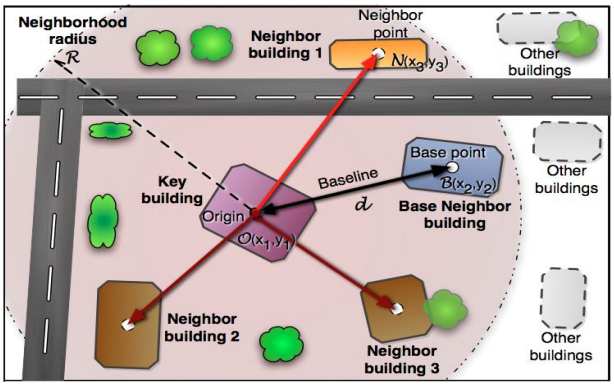


Fig. 2 Building hashing scheme. Hashing baseline is formed by the key building and base neigbor centers.

Algorithm 1 – Hashing

Input: buildings
hash_table ← []
for all *key_building* in *buildings* **do**
  (x_1,y_1) ← centroid(*key_building*)
  a_1 ← area(*key_building*)
  N ← neighbors(*key_building, max_neighbors, R*)
  for all *base_neighbor* in *N* **do**
    (x_2,y_2) ← centroid(*base_neighbor*)
    a_2 ← area(*base_neighbor*)
    baseline ← vector((x_2, y_2), (x_1, y_1))
    other_neighbors ← *N* – {*base_neighbor*}
    for all *neighbor* in *other_neighbors* **do**
      (x_3, y_3) ← centroid(*neighbor*)
      a_3 ← area(*neighbor*)
      (f_x, f_y) ← normalize((x_3,y_3), *baseline*)
      (q_x, q_y) ← quantize(f_x, f_y)
      q_{a13} ← quantize(a_1 / a_3), q_{a23} ← quantize(a_2 / a_3)
      hash_key ← [q_x, q_y, q_{a13}, q_{a23}]
      value ← [*key_building, base_neighbor*]
      hash_table[*hash_key*].append(*value*)
return: *hash_table*

Here, we are proposing a building detection algorithm to extract building footprints from a given image (*e.g.* Fig. 4 (a)) even in the presence of occlusion from vegetation and similarities in the appearances of roads and building roofs. As a first step, we use an image classifier to obtain pixel-wise classification probabilities for buildings (Fig. 4 (c)). The classifier is a three-class Random Forest classifier trained over training samples of vegetation, road and building regions. Building classification results are binarized and noise reduction filters are applied to remove unwanted salt and pepper noise and small components. Each connected component in the resulting image that has a large enough area and that is structurally not too thin for a building is considered as a building seed region. Holes are filled and building seeds are expanded outwards based on color similarity to complete missing parts of the buildings. Final detection results are shown in Fig. 4 (d). Missed detections and false detections are highlighted in Fig. 4 (f). Most of the missed detections are due to tree occlusions and buildings with similar appearance to roads. False detections occur when isolated road patches are detected as buildings. With our method we obtain satisfactory results and the detection accuracy is listed in results section.

2) Hierarchical Querying

We localize the query image by employing a hierarchical query approach. We use the buildings extracted from the query image and the previously built hash table. As detailed in Algorithm 2, we first independently localize every detected building in the query image.

In this step, building neighborhoods are generated from the detected buildings. A similar baseline and key generation procedure is followed to generate a query key to the hash table. All the values in the hash table that correspond to the generated keys are candidate buildings for the query building. Using the key building and baseline neighbor numbers in each table entry, probable orientation and scales votes for each candidate are generated. The output of each single building localization step is the index of the best matching building on the map as well as scale and orientation hypotheses for the query image.

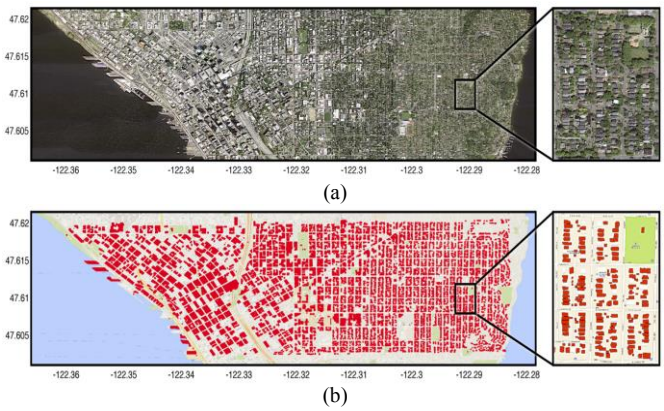
We then combine the single building localization results that agree on image center locations to obtain final localization results as explained in Algorithm 3. We cast votes for image orientation and scale and location of the image center. Votes are quantized in order to compensate for mismatches due to individual localization errors and inaccuracies. The highest voted area is the localization estimate for the query image center. An improvement step can also be applied to this quantized localization estimate to further fine-tune it and get more precise localization results. Our hierarchical approach enables the method to be robust against missing or false building detections and possible incorrect localizations of individual buildings.

During testing we take the area with the maximum number of votes as the estimated location of the query image. This is a hard localization decision. It is also possible to make a soft localization decision, which would examine the top k voted locations within the large-scale map. Later, these candidate regions can be examined more closely.

IV. LARGE-SCALE BUILDING DATASET

In order to build our initial semantic hash table, we used a building dataset of Seattle provided by Seattle City GIS Program [20]. In the provided GIS dataset, geographic coordinates and detailed outline contours of buildings in Seattle are given. In the dataset, buildings are extracted from imagery acquired in 2009 and further hand processed for higher precision, in the order of centimeters. Data provided is composed of individual building contour polygons with WGS84 and feet coordinates. Whole building footprint data consists of 284,017 buildings from Seattle, covering an area of approximately 23 km x 38 km.

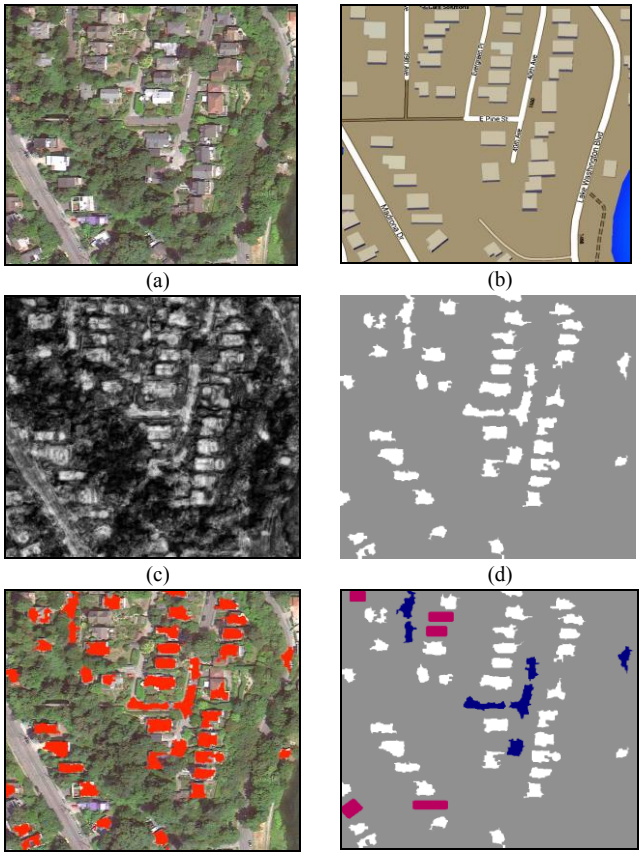
The part of the dataset we used in our experiments covers a dense 16.5 km^2 map area that contains 7,000 buildings. We obtained satellite images of the area from Google Maps, which is shown in Fig. 3 (a). GIS building outline data overlaid on top of Google Maps images is shown in Fig. 3 (b). We chose this part of the city to include both the downtown area containing large and sparse buildings (Fig. 3 West) and the residential Central District area containing neighborhoods of small and dense buildings (Fig. 3 East).



(a)

(b)

Fig. 3 (a) Google Maps Satellite image of the area that has been used in the localization queries. (b) Same area with ground truth GIS building outlines ovelayed on top of the Google Maps image of the area. All 7,000 buildings with their outlines are shown on the map. Best viewed on computer screen and in color.



(a) (b) (c) (d) (e) (f)

Fig. 4 Building detection results on a sample tile. (a) Google Maps Satellite image tile of size 1024 x 1024 px. (b) Google Maps tile. (c) Pixel-based building classification results obtained from satellite image, brighter values are higher building probabilies. (d) Final building detection results. (e) Building detection results overlayed on satellite image. (f) Building detection results showing correct (white), missing (magenta) and false (dark blue) detections. Best viewed on computer screen and in color.

Algorithm 2 – Locate Single Building

Input: key_building, hash_table
$(x_1,y_1) \leftarrow$ centroid(*key_building*)
$a_1 \leftarrow$ area(*key_building*)
$N \leftarrow$ neighbors(*key_building, max_neighbors, R*)
query_hash_table ← []
for all *base_neighbor* in *N* **do**
  $(x_2,y_2) \leftarrow$ centroid(*base_neighbor*)
  $a_2 \leftarrow$ area(*base_neighbor*)
  baseline ← vector((x_2, y_2), (x_1, y_1))
  other_neighbors ← *N* – {*base_neighbor*}
  for all *neighbor* in *other_neighbors* **do**
    $(x_3, y_3) \leftarrow$ centroid(*neighbor*)
    $a_3 \leftarrow$ area(*neighbor*)
    $p \leftarrow [(x_1, y_1), (x_2, y_2), (x_3, y_3)]$
    $(f_x, f_y) \leftarrow$ normalize((x_3,y_3), *baseline*)
    $(q_x, q_y) \leftarrow$ quantize(f_x, f_y)
    $q_{a13} \leftarrow$ quantize(a_1 / a_3), $q_{a23} \leftarrow$ quantize(a_2 / a_3)
    hash_key ← [q_x, q_y, q_{a13}, q_{a23}]
    values ← *hash_table*[*hash_key*]
    for all *value* in *values* **do**
      baseline ← *value*
      building_id ← *value*[0]
      building_theta ← calculate_theta(*baseline, p*)
      building_scale ← calculate_scale(*baseline, p*)
      $q_{theta} \leftarrow$ quantize(*building_theta*)
      $q_{scale} \leftarrow$ quantize(*building_scale*)
      building_key ← [*building_id*, q_{theta}, q_{scale}]
      query_hash_table[*building_key*]++
match ← {scale, theta, id} of max(*query_hash_table*)
return: *match*

Algorithm 3 – Image Query

Input: query_image, hash_table
image_center_votes ← []
query_buildings ← detect_buildings(*query_image*)
for all *key_building* in *query_buildings* **do**
  match ← **LocateSingleBuilding**(key_*building, hash_table*)
  image_center_vote ← calculate_image_center(*match*)
  $q_{center} \leftarrow$ quantize(*image_center_vote*)
  image_center_votes[q_{center}]++
best_center ← {x, y} of max(*image_center_votes*)
return: *best_center*

V. Experiments and Results

To test the semantic geometric hashing method presented, building contour data from Seattle GIS building data is used to construct the hash table and locations of satellite image patches are queried. Non-overlapping tiles from Google Maps satellite images with approximately 40 cm/px resolution are used for building detection and querying for localization. The test area with 7,000 buildings in Fig. 3 is fully covered with 80 image tiles of size 1024x1024 px. We report on the accuracy of the building detection and localization stages and analyze the localization step under different parameter configurations.

A. Building Detection Results

Building detection has **65% *precision*** and **81% *recall***, based on building locations. Method has high recall, but also suffers from false positives that decrease the precision. Majority of false positives come from roads having similar appearance to rooftops and false negatives come from tree-occluded buildings and complex building structures.

B. Large-Scale Localization Results

Considering the tiles that have more than 80 buildings and 18 neighbors for each model, the overall localization accuracy is **91%**. In Fig. 5 we further investigate the effect of neighborhood size. In Fig. 5 we compare localization results of detected buildings and ground truth building locations. Difference does not only come from weaknesses of building detection algorithm, but also from building differences between 2009 GIS information and 2014 satellite images. Even with these problems, the method performs robustly and accurately localizes the query image.

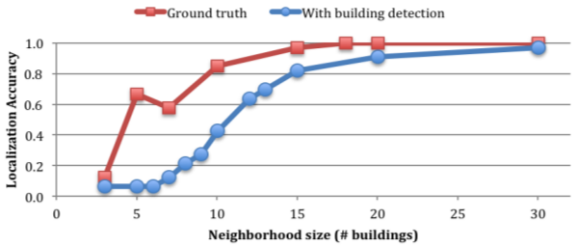


Fig. 5 Localization accuracy vs. model neighborhood size. Accuracy increases as the number of buildings increase.

In Fig. 6 we report localization accuracy divided into *denseness* categories of tiles, where residential areas are denser than downtown areas. It can be observed that as denseness increases, localization accuracy increases. This is due to dense regions providing more semantic information and also building detection algorithm working better in these dense regions with smaller and rectangular buildings.

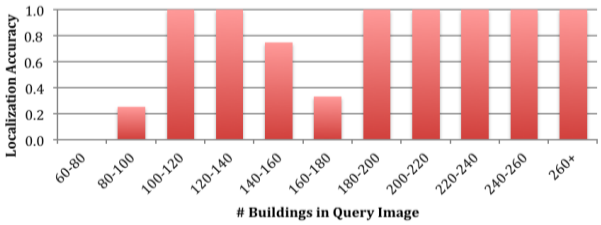


Fig. 6 Localization accuracy vs number of buildings in the query image. Accuracy increases as the number of buildings increase.

We also explored how much accuracy we get if we look in to top *k* votes locations instead of picking the maximum voted location. Fig. 7 shows accuracy vs. number of top locations considered, for different neighborhoods. We see that it is highly probable to find the correct location if top 4 locations are investigated as opposed to using maximum vote location.

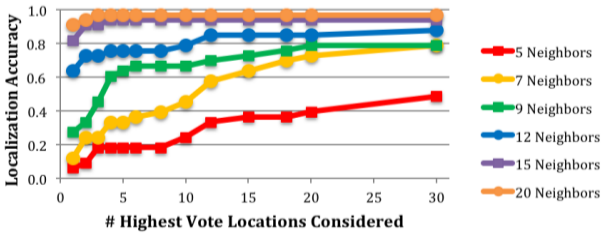


Fig. 7 Localization accuracy vs. model neighborhood size. Accuracy increases as the number of buildings increase.

C. Performance Evaluation and Scalability Results

To evaluate the presented approach from different performance aspects, we performed various experiments on our large-scale dataset. For the experiments we used 7,000 query buildings. Fig. 8 (a) shows that the accuracy of our method is not affected adversely when additional buildings are added to the hash-table, hence proves an aspect of the scalability of the method. In Fig. 8 (b) we show the rotation invariance of the results by using extreme cases of 90, 180 and 270-degree rotations, which was already expected from the construction of the method. Fig. 8 (c) and (d) show respectively the linear and quadratic relations of total number of buildings and number of neighbor buildings to the hash-table size, again showing the scalability of the method. In this test we hashed up to 250,000 buildings on a single 2.2 GHz Intel i7 computer with 16 GB memory, where hash table size reached up to 6.5 GBs.

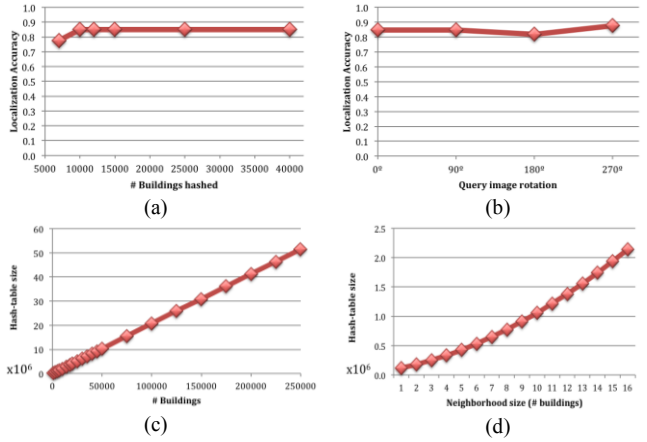


Fig. 8 (a) Localization accuracy vs. number of buildings hashed (queried with 7,000 buildings fixed). (b) Localization accuracy vs. rotation, (c) Total hash-table entries for number of buildings hashed. (d) Total hash-table entries vs. model neighborhood size.

VI. Conclusion

We presented a complete vision-based instantaneous localization pipeline that has no manual steps, where we can localize a given single aerial image in a very large map. For semantic localization, we presented a novel hierarchical semantic hashing scheme that is invariant to scale and rotation and robust to occlusions and missing data. Our method scales linearly with increasing number of buildings stored in the database. Our approach can be modified to work with arbitrary top view images and with other semantic landmark information in addition to buildings. Results for using our method to perform efficient localization of satellite images on large-scale GIS maps are presented. For experiments in an urban area with large number of buildings, method achieves high localization accuracy as long as sufficient buildings exist in the query image. Furthermore, we presented an image-processing algorithm to detect buildings from a satellite image that completes our vision based localization pipeline.

References

[1] D. Filliat, "A visual bag of words method for interactive qualitative localization and mapping," presented at the IEEE International Conference on Robotics and Automation (ICRA), 2007, pp. 3921–3926.

[2] J. P. Lewis, "Fast normalized cross-correlation," *Vision interface*, vol. 10, no. 1, 1995.

[3] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Computing Surveys (CSUR)*, vol. 40, no. 2, pp. 1–60, Apr. 2008.

[4] A. Pronobis, "Semantic mapping with mobile robots," 2011.

[5] W. Jiang, K. L. Chan, M. Li, and H. Zhang, "Mapping low-level features to high-level semantic concepts in region-based image retrieval," presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005, vol. 2, pp. 244–249.

[6] A. R. Zamir and M. Shah, "Accurate image localization based on Google maps street view," presented at the European Conference on Computer Vision (ECCV), Berlin Heidelberg, 2010, pp. 255–268.

[7] T. Senlet and A. Elgammal, "A framework for global vehicle localization using stereo images and satellite and road maps," presented at the IEEE International Conference on Computer Vision Workshops (ICCV-CVVT), Barcelona, 2011, pp. 2034–2041.

[8] C. Wu, F. Fraundorfer, J.-M. Frahm, and J. Snoeyink, "Image localization in satellite imagery with feature-based indexing," presented at the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS), Beijing, 2008, pp. 197–202.

[9] Y. Lin and G. Medioni, "Map-enhanced UAV image sequence registration and synchronization of multiple image sequences," presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–7.

[10] M. W. Achtelik, S. Lynen, S. Weiss, L. Kneip, M. Chli, and R. Siegwart, "Visual-inertial SLAM for a small helicopter in large outdoor environments," presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012, pp. 2651–2652.

[11] Y. B. Yang and H. T. Tsui, "Mobile robot localization by geometric hashing and model-based scene matching," presented at the International Conference on Pattern Recognition (ICPR), 1996, vol. 1, pp. 181–185.

[12] T. J. Cham, A. Ciptadi, W. C. Tan, M. T. Pham, and L. T. Chia, "Estimating camera pose from a single urban ground-view omnidirectional image and a 2D building outline map," presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 366–373.

[13] I. Rigoutsos and H. J. Wolfson, "Geometric Hashing," *Computational Science & Engineering, IEEE*, vol. 4, no. 4, p. 9, 1997.

[14] H. Van Dijck, M. Korsten, and F. Van Der Heijden, "Robust 3-dimensional object recognition using stereo vision and geometric hashing," presented at the International Conference on Image Processing (ICPR), 1996, vol. 1, pp. 329–332.

[15] J.-J. Liu and R. Hummel, "Geometric hashing with attributed features," presented at the CAD-Based Vision Workshop, 1994, pp. 9–16.

[16] F. C. Tsai, "A probabilistic approach to geometric hashing using line features," *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 182–195, 1996.

[17] B. P. Olsen, "Automatic change detection for validation of digital map databases," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, pp. 569–574, 2004.

[18] N. Ekhtari, M. R. Sahebi, M. V. Zoej, and A. Mohammadzadeh, "Automatic building detection from Lidar point cloud data," presented at the International Arch. of the Photogrammetry, Remote Sens. and Spatial Information Sciences (ISPRS), Beijing, 2008, vol. XXXVII, part B4.

[19] N. Shorter and T. Kasparis, "Automatic vegetation identification and building detection from a single nadir aerial image," *Remote Sensing*, vol. 1, no. 4, pp. 731–757, 2009.

[20] *Seattle's Data Site*. [Online]. Available: https://data.seattle.gov/dataset/2009-Building-Outlines/y7u8-vad7. [Accessed: 19-Dec-2013].