

# Aerial image sequence geolocalization with road traffic as invariant feature

Gellért Mátyus

German Aerospace Center, Germany, Remote Sensing Technology Institute  
gellert.mattyus@dlr.de

Friedrich Fraundorfer

German Aerospace Center, Germany, Remote Sensing Technology Institute  
Graz University of Technology, Austria, Institute for Computer Graphics and Vision  
fraundorfer@icg.tugraz.at

## Abstract

*The geolocalization of aerial images is important for extracting geospatial information (e.g. the position of buildings, streets, cars, etc.) and for creating maps. The standard is to use an expensive aerial imaging system equipped with an accurate GPS and IMU and/or do laborious Ground Control Point measurements. In this paper we present a novel method to recognize the geolocation of aerial images automatically without any GPS or IMU. We extract road segments in the image sequence by detecting and tracking cars. We search in a database created from a road network map for the best matches between the road database and the extracted road segments. Geometric hashing is used to retrieve a shortlist of matches. The matches in the shortlist are ranked by a verification process. The highest scoring match gives the location and orientation of the images. We show in the experiments that our method can correctly geolocalize the aerial images in various scenes: e.g. urban, suburban, rural with motorway. Beside the current images only the road map is needed over the search area. We can search an area of 22500 km<sup>2</sup> containing 32000 km of streets within minutes on a single cpu.*

## 1. Introduction

It is required for numerous applications that images and videos are tagged with their location on the earth's surface (geolocation). The geolocalization of aerial images is particularly essential for extracting geospatial informations (e.g. the position of buildings, streets, cars, etc.) and for creating maps. The aerial imaging systems typically include an accurate (and expensive) GPS and IMU. By avoiding these instruments a simple consumer camera could also be used to create georeferenced aerial images. People would be able to use their own camera during leisure and touristic activities (e.g. hang gliding, air balloon flight, sightseeing flight, glider flight, etc.) to create georeferenced and orthorectified images. A crowd-sourced database of orthorectified aerial images (similar as OpenStreetMap for maps) could be used for mapping applications.

Although nowadays even consumer grade cameras can provide geotagging, it might be needed to find the geolocation of the camera, image and the objects in the scene based only on the visual information. (See the IARPA Finder program <sup>1</sup>). Even with a GPS position tag the orientation of image is usually still unknown. Aerial image camera systems equipped with a very precise GPS and IMU also might lack the geolocation in case of an outage. The automatic geolocalization of these images can reduce expenses by avoiding a new flight or laborous manual work. It is also getting common to acquire images from a unmanned aerial vehicle (UAV). These can only carry light payloads, limiting the accuracy of the onboard GPS and IMU. Retrieving a more accurate image position and orientation during post processing could be needed.

UAVs also need a backup localization system in case of an outage of the GPS (GNSS). A GPS outage can happen either due to a problem in the device or because of external jamming of the GPS signal. The jamming of the GPS signal is a realistic threat, not only in military environments. The problem of GPS jamming is addressed in research, e.g. in [11].

Our goal is the geolocalization of aerial photo sequences on a large scale by utilizing only the image information and the road network. We propose to match the information acquired from the current aerial image sequence to an object database

---

<sup>1</sup><http://www.iarpa.gov/Programs/ia/Finder/finder.html>

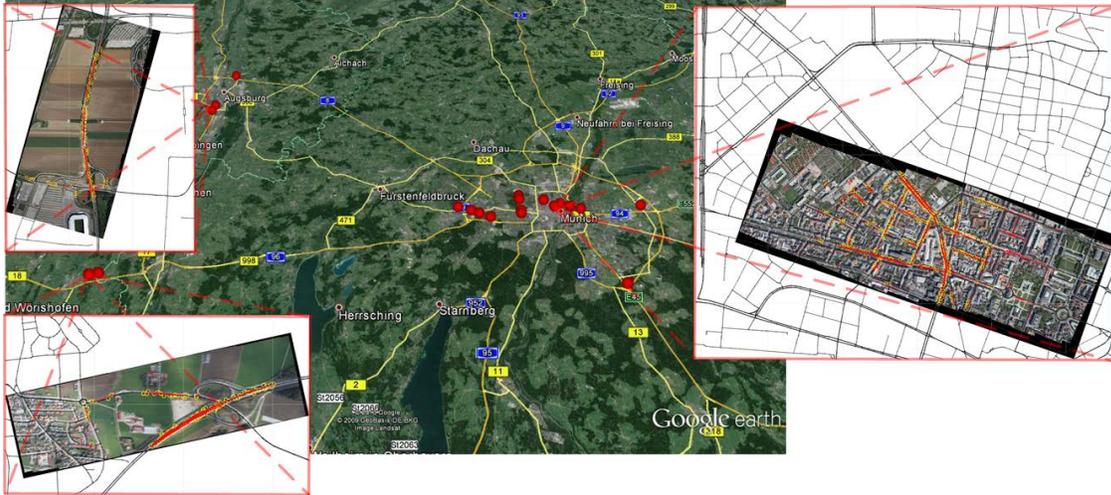


Figure 1. Illustration. We search the geolocation of aerial image sequences. The red dots on the map show the ground truth positions. The photos can be located by matching the road traffic (in red) in the image to the road network (in black). The matching tracks are marked with yellow circles in the images. The search area can be as large as the shown map. The accuracy of the location is around 25 m.

invariant to the lighting and weather conditions, the road network map. Since the maintenance of an accurate, up-to-date road map is needed for many other applications (e.g. navigation, administration, etc.), it is more easily accessible than a large image database and requires significantly less storage <sup>2</sup>.

We detect the road traffic in the image scene by tracking cars over the frames. By assuming the cars drive on the roads the vehicle trajectories can be interpreted as subsets of the roads, and be matched to the road map.

We propose a method for the fast retrieval of a shortlist containing the possible correct location. This retrieval is based on Geometric Hashing. We call it *Polyline Based Geometric Hashing (PLBGHashing)*, and it can search rapidly over a larger search area. The more complex verification matching needs to be done only on the retrieved shortlist. The car tracks can be considered as a road detector with low completeness but high correctness. We combine this with a simple pixel color based road detector (with high completeness but low correctness) for the verification.

We analyzed the proposed *PLBGHashing* on synthetic data generated from the road network of two large cities. This confirmed that the pattern of the road network is discriminative on a larger scale. An evaluation was done on 20 image sequences captured over urban, suburban, rural with motorway, and industrial scenes by a consumer-grade camera mounted on an airplane, the Figure 1 shows the location of the scenes. In this test the *PLBGHashing* provided the shortlist, while the verification ranked the correct geolocation as the best match in most of the cases.

Our main contributions are: (1) We use car tracks to detect parts of the road network. This does not need a known ground sampling distance (GSD<sup>3</sup>) as standard road detectors. (2) We utilize road networks as appearance invariant features to localize an aerial image sequence over a large area. This also avoids the need for an image database over the search area. (3) We present a geometric hashing method to match partial line segments to line structures. This does not need a complete road detector which detects relations between roads as a graph, e.g. intersections.

## 2. Related work

The standard for creating georeferenced aerial images is using an accurate GPS and IMU and/or measuring Ground Control Points manually [15]. Other approaches utilize the idea to localize an image from only the image information also on a large scale. The queried image can be matched to a huge pool of landmark photos [32], while the 3D building information might also be utilized [3]. Li et al. [17] determine the absolute (world) camera pose using 3D point clouds. Mueller et al. [21] improve the absolute geometric accuracy of satellite images by automatically extracting Ground Control Points from existing orthorectified reference images. In Wu et al. [30] satellite images are localized with feature-based indexing, but not on a real large scale (maximal 16 km × 12 km). In Lin and Medioni [19] UAV images are matched to georeferenced ortho images (map). But the search area is limited by manually labeling several correspondences between the first frame of the

<sup>2</sup>The OpenStreetMap is an open, crowd-sourced map with good coverage and fast updates.

<sup>3</sup>Ground Sample Distance, the distance between pixel centers measured on the ground.

UAV sequences and their corresponding satellite images.

All these methods require an already existing image database, and capturing and maintaining this is laborious. Using crowd-sourced image databases (e.g. *Flickr*, *Google Picasa*) this difficulty can be overcome. However, the landmark images are unevenly distributed and biased to touristic highlights. The work Lin et al. [18] addresses this problem by matching terrestrial image to remote sensing data, which also provide coverage over locations where no crowd-sourced data are available.

A new direction is the usage of *non-image* databases. Baatz et al. [4] employ the terrain map to geolocalize images from the mountain silhouette in the Alps.

Brubaker et al. [8] utilize the road network and visual odometry to locate a driving vehicle without any other information. They show that a long track on the road network is characteristic enough to obtain the geolocation of the car. Our method shares the idea of exploiting the pattern of the roads in the form of car tracks, but instead of a single long vehicle trajectory we use many short ones.

In Konzempel and Reulke [14] aerial images are orthorectified (projected on the Earth’s surface) without an IMU. The orientation is initialized by an accurate GPS measurement and optimized by matching the detected streets in the image to the road network. In comparison to this method, our approach does not require an accurate position information, but only a search area, which might be as large as an entire metropolitan area.

In [25], road networks are represented as graphs, with road intersections as graph vertices. It is assumed that the intersections are detected correctly and the georegistration problem can be solved as a graph matching problem. In comparison to this work, in our approach the road intersections does not need to be detected in the image. We assume that there is no appropriate intersection detector or there might be no intersections in the image (just non intersecting roads).

In Wilson and Hancock [27] the graph structure of the road network (junctions and roads between them) in aerial image is extracted using a relaxational line-finder. Then the graph junctions are matched to junctions in the map using probabilistic relaxation. This method assumes that road junctions are present and they can be detected. In contrast, our method works without junctions, it needs only segments of the road network and we consider a much larger search area.

Li et al. [16] extract and match line segments instead of junctions. They define rotation and translation invariant features between the segments and use these pairwise features to calculate the cost of an assignment between aerial image and map. This combinatorial optimization problem is solved via continuous relaxation labeling. Solving this optimization problem is non trivial, specially for a large number of variables. In [16], the orientation of the image is used to limit the possible matches. In our problem we do not have access to this orientation information. We provide a more detailed analysis on this method in section 4.2.3.

Gros et al. [10] use local, geometric invariant features to match images related by similarity or affine transformations. For similarity transformations they compute geometric invariants from pairs of line segments having an endpoint in common. The invariants are the angle between the two segments and the ratio of the length of the segments. These invariants are matched between the images, similarity transformations are computed from the matches and they are aggregated in a Hough-transform manner, i.e. clusters are searched in the parameter space. In contrast to this method, in our problem the segment length ratio invariant can not be applied. A track segment can lay anywhere within the map segment. Additionally, our tracks are mostly straight (or with a small curvature), the angle between the consecutive line segments is usually around zero and thus the angle invariant is not discriminative enough.

The COCOA system [1] addressed the tracking of moving objects in aerial videos. Xiao et al. [31] track cars in wide field of view aerial videos, however they need already georeferenced images to leverage traffic flow priors from road maps.

### 3. Image sequence to road database matching

We extract the road traffic from the images and match the car tracks to the road network. Detecting the roads directly would be more straightforward, but this may pose problems due to the high intraclass variety of roads. The methods described in [26], [20] show good results, but they work on images with a constant GSD and the effect of an unknown GSD is not investigated. In our case there is no information about the GSD. Using only the car tracks also highlights that already a fraction of the road network is enough for the geolocalization.

The intraclass variety in visual appearance of cars is much smaller than that of roads. Thus existing robust object detection methods (e.g. Viola-Jones object detection framework [24]) can be applied with satisfying results (e.g. a detector based on boosting is presented in [12]). The features of the car detector are pixel intensity differences, which can be robust against different weather and lighting conditions. The motion information is also a strong cue, enabling object detection in cases where a non-moving object could not be distinguished from the background. We utilize the motion information by using the tracks of only the moving cars to reduce the possible false positive car detections. The length of the tracked cars in the image

also gives information about the GSD. The proposed geolocalization method could also work with a suitable road detector (e.g. based on deep learning), or by labeling the roads manually and limiting the GSD search space.

### 3.1. Track extraction

The track extraction works on a mosaic image compiled from the single images. We use the tool *VisualSFM* [29] to calculate the camera parameters and 3D point coordinates in the scene. The earth surface is assumed a plane, and a homography is calculated between the images from the camera parameters and the equation of the plane.

We detect the cars independently on the image frames. We apply the Viola-Jones object detection framework [24] with the Gentle AdaBoost [9]. The detector is rotation variant, thus we rotate the image in steps to cover all directions and group the independent detections together (it would be more time efficient to use a multi-view detector).

The detections are transformed to the mosaic image. During the tracking we model the motion of the cars with a simple linear model and use Kalman-filtering. In each frame the tracked objects are assigned to detections matching the predicted object positions. We discard the short tracks to avoid false positives. This is a simple method, delivering satisfying results.

### 3.2. Matching the tracks to the road network

We formulate the aerial image geolocalization as a model recognition and pose estimation task of the road network pattern. The search area is tiled to overlapping square areas (in object recognition terms a tile is a model). An image scene query returns a model and a transformation between the image and the model. Since the absolute world position of the tile (model) is known, the transformation from the image coordinates to the geocoordinates can be recovered. The recognition has to handle geometric transformations for the pose estimation, a high number of models ( $> 10000$ ) for larger search areas, and partially occlusions since the extracted tracks are just a subset of the road network. The geometric hashing method is suitable for handling these challenges.

#### 3.2.1 Geometric hashing

Geometric hashing is an efficient, low polynomial complexity technique for matching geometric features against a databases of such features. Matching is possible even when the recognizable database objects have undergone transformations or when only partial information is present. We describe briefly the geometric hashing defined for  $2D$  feature points. For a more detailed explanation the reader is referred to [28]. In the preprocessing step of geometric hashing a hash table is generated:

1. Calculate the feature points of the model.
2. For each ordered point pair (basis) calculate the remaining points in the coordinate frame defined by the basis.
3. For each remaining point coordinate in the basis frame quantize the coordinates to a grid. Use the grid coordinate as hash index and insert (into the hash table) an entry containing the identifier of the model and the basis used to generate the hash index.

The matching is done with the following steps:

1. Calculate the feature points in the image to be matched.
2. For an arbitrary selected ordered point pair (basis) calculate the remaining points in the coordinate frame defined by the basis.
3. For each remaining point coordinate in the basis frame quantize the coordinates to a grid. Use the grid coordinate as hash index and cast a vote for all the models and the bases present at the hash index.
4. The models and bases with the most votes are match candidates. Select a candidate, recover the transformation between the query image and the reference model (based on the basis correspondence) and verify the transformation. If the verification fails, repeat this step (Step 4) with the next candidate.

Nakai et al. [22] were able to retrieve document images from a database of several thousand images in a fraction of a second. In Stein and Medioni [23] a curvature based geometric hashing is presented. In our case the most important feature is not the shape of the individual roads but the distances between the different roads, therefore we employ a geometric hashing similar as in [28].

**Map and track representation** The roads in the map are represented as 2D polylines. A polyline contains vertices (points), and the road consist of the line segments between consecutive vertices. The tracks are also polylines, whereas the vertices are the positions in the registered image during different frames. Although the representation of the tracks and roads is identical, there is no strict correspondence between the vertex positions in the road network and the tracks. A track segment can lie anywhere within the road segment.

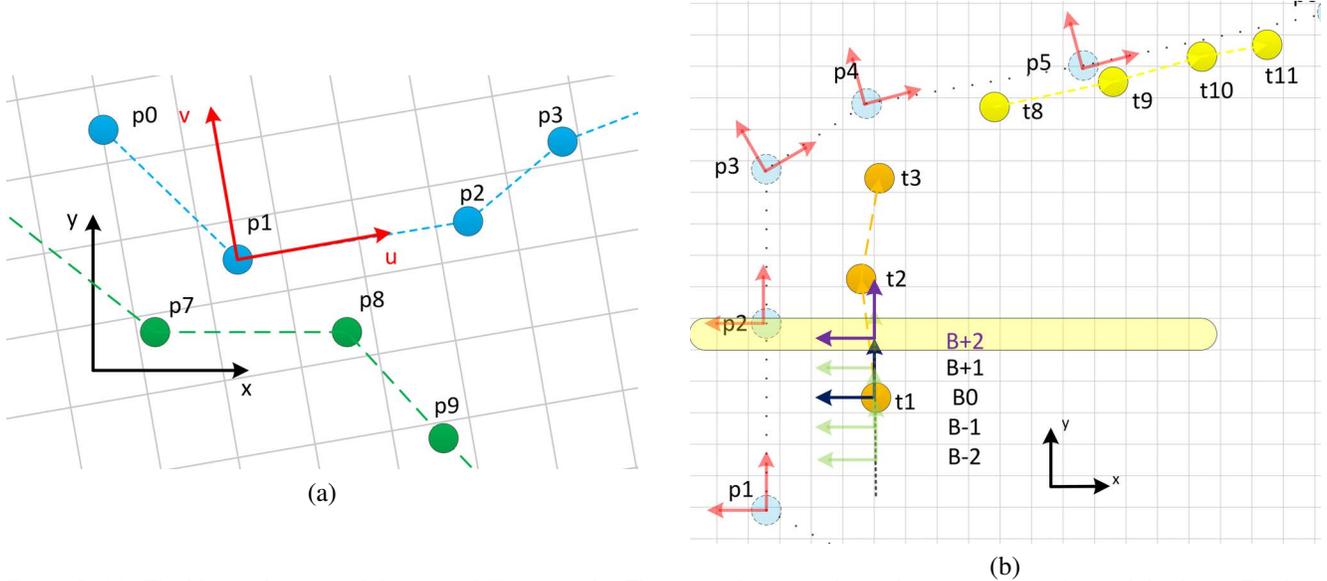


Figure 2. (a): The blue and green polylines are different roads. The x-y is the original coordinate system of the road database. The line segment p1-p2 defines the basis (u,v) with the center point and the direction. The grid points intersected by line segments are the indices for the hash table. (b): The p points are the road vertices (from the database), the t points are the track vertices extracted in the mosaic image (with different color for different cars). The red arrows are the road bases in the hash table. The basis B0 from the track end points t1-t3 is shifted along the track. At B+2 position the track basis lies in the the same grid row, as the road basis.

**Polyline Based Geometric Hashing** The *PLBGHashing* method compiles the polylines to the hash table. Since we assume the earth’s surface to be a plane and the tracks projected on this plane, the perspective transformation between the tracks and the roads can be reduced to a scaling, rotation and translation. As a basis we use a segment in a form of one point and a direction instead of two or more points. The relation of a  $\mathbf{p}$  point’s original coordinate  $(x, y)$  and the coordinate  $(u, v)$  in the basis defined by a line segment  $S$  with end points  $\mathbf{p}_{s_1}, \mathbf{p}_{s_2}$  is described by the equation:

$$\mathbf{p} - \mathbf{p}_{s_1} = u\mathbf{p}_x^s + v\mathbf{p}_y^s \tag{1}$$

where  $\mathbf{p}_x^s = \frac{\mathbf{p}_{s_1} - \mathbf{p}_{s_2}}{\|\mathbf{p}_{s_1} - \mathbf{p}_{s_2}\|}$  and  $\mathbf{p}_y^s$  is  $\mathbf{p}_x^s$  rotated by  $90^\circ$ . On this basis the 2D hash table is invariant to rotation and translation but not to the scaling. To find the correct GSD we query with multiple scales. By leveraging the low variation of the car lengths (4 – 6m), the number of different scales are kept low. We calculate the maximum in the histogram of the car lengths (the bounding rectangle delivered by the detector) in the tracks. Utilizing this length we needed only 3 scales to search in our experiments to get good geolocalization.

Each hash bin stores a set of entries. An entry contains a model number and a basis. It is important to store a set with unique elements, otherwise some bins could be biased by areas containing roads densely.

**Database construction (off-line)** The search area is tiled to squares with a fixed side length  $d_s$  and overlap. The nodes are interpolated to make the line segments shorter as a defined maximal value  $d_{max}$ . Then the hash table is generated by the following steps:

For each polyline (road)  $P_j$  in each tile  $M_i$  do:

1. For each line segment  $S_k$  in the polyline  $P_j$ :

- (a) Create an entry  $E_k$  with the model number  $i$  and the basis of  $S_k$ .
- (b) Solve the (Eq. 1) with  $S_k$  to calculate  $(u_m, v_m)$  of all the vertex points in  $M_i$ .
- (c) Quantize the points  $(u_m, v_m)$  to a grid  $(u_m^q, v_m^q)$ .
- (d) Calculate the coordinates  $(u_l^q, v_l^q)$  of the rasterized line in the grid between the neighboring  $(u_{m-1}^q, v_{m-1}^q), (u_m^q, v_m^q)$  vertices. The  $(u_l^q, v_l^q)$  line points are indices for the 2D hash table. For each line point  $(u_l^q, v_l^q)$  add the entry  $E_k$  to the set at the  $(u_l^q, v_l^q)$  hash bin.

The Figure 2a illustrates the hash index generation. The complexity of the number of bases is  $O(n)$ , where  $n$  is the number of node points after interpolation in the tile. The complexity of hash table entries per basis is  $O(l/d_g)$  where  $l$  is the sum of road length and  $d_g$  is the grid size for geometric hashing. Since  $d_g$  is fixed in an implementation the complexity of hash entries is  $O(nl)$ .

**Shortlist retrieval (on-line)** To find the correct geolocation of a scene, a basis (1-point and direction) in the tracks has to be aligned to a similar basis in the correct model with the correct scale. The direction of a road segment can be properly recovered from the end points of a track. We choose long straight tracks, because they define the direction more accurately if the vertex positions are noisy. The correct scale is searched with multiple queries. The list of scales is derived from the car length. Since the line segment length is limited by  $d_{max}$ , a street vertex exists in the range  $[-d_{max}/2, d_{max}/2]$  along a track vertex, the Figure 2b illustrates this. It is enough to search this range in shifts by the grid size  $d_g$ , because then the hash indices are the same at creation and query. We resize the scene with all the search scales. For each resized scene we choose track segments  $T_i$  (the line segment between the end points). For each  $T_i$  in both directions we do:

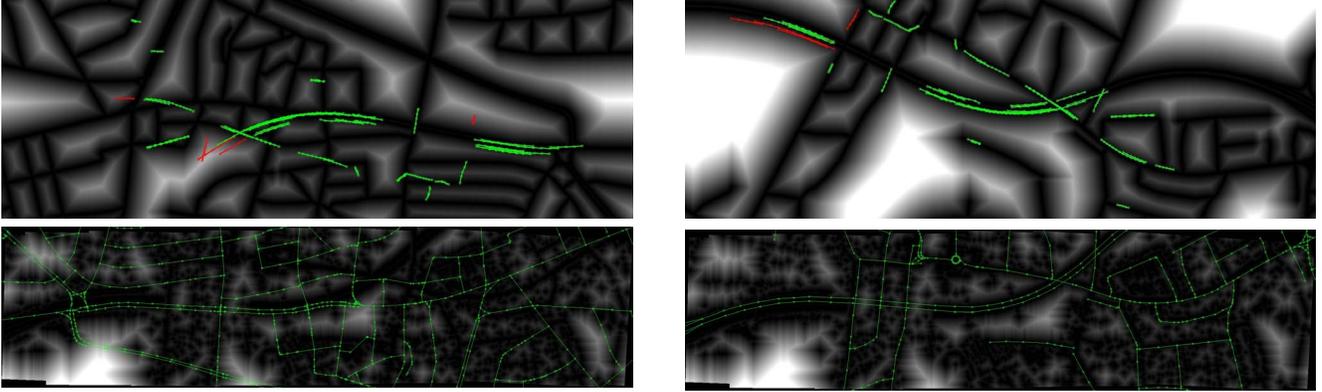
1. Shift  $T_i$  between  $[-d_{max}/2, d_{max}/2]$  in  $d_g$  steps.
2. Each shift gives a line segment  $S_k$ . For each  $S_k$ :
  - (a) Solve the (Eq. 1) with  $S_k$  to calculate  $(u_m, v_m)$  of the vertex points of all the tracks in the scene.
  - (b) For each polyline (track)  $P_i$  in the scene:
    - i. Calculate the rasterized points  $(u_m^q, v_m^q)$  of the vertices
    - ii. Calculate the coordinates  $(u_l^q, v_l^q)$  of the discrete (rasterized) line segment in the grid between the  $(u_{m-1}^q, v_{m-1}^q), (u_m^q, v_m^q)$  points.
    - iii. Each point  $(u_l^q, v_l^q)$  of the line segment is an index for the hash table. For every entry at the hash bin  $(u_l^q, v_l^q)$  cast a vote for a recognition (model, basis and scale).

After the query with multiple straight tracks, the votes are aggregated. One recognition gets only one vote from a bin, otherwise a basis might be biased due to many tracks over the same index. A shortlist of matches is created by sorting the recognitions based on the number of received votes. The transformation from image to a world coordinate can be recovered from the basis in the image and in the model, the scale and the world position of the model (tile).

### 3.3. Verification of the shortlist

The retrieved shortlist also contains false recognitions. We re-sort the shortlist by a stricter criteria, the verification match score  $S$ . The quality of the match is measured by the mean distance  $d_{m_t}$  of the tracks  $R_t$  to the roads in the model, like the Chamfer Matching [5]. The distance image  $I_d$  is calculated to the roads [7], and the  $I_d$  pixels along the tracks are extracted. Since the tracks may contain outliers, we calculate a trimmed mean, where the 10% of the tracks with the highest distances are discarded. The  $d_{m_t}$  does not penalize roads not covered by tracks. To compensate for this, we extract a second road detector with high completeness but low correctness. We assume that the color of the roads is similar all over the scene. The CIELUV color values of the image pixels are grouped to 16 clusters by k-means. Clusters are added to the road class until 85% of the tracks lie over pixels belonging to the road class. A second  $d_{m_p}$  mean distance is calculated as the distance of the roads in the model to the roads detected by the pixel based detector ( $R_p$ ). The Figure 3 shows the distance images, a brighter pixel is a larger distance.

We assign likelihood values  $L_t$  ( $R_t$  match), and  $L_p$  ( $R_p$  match) to the two mean distances  $d_{m_t}$  and  $d_{m_p}$ . The likelihood function  $f(x)$  of  $d_{m_i}$  for a correct match are modeled by the following function, a normal distribution function  $\mathcal{N}(\mu, \sigma)$  if  $x \geq \mu$ , uniform with the max of the normal distribution if  $0 \leq x < \mu$  and 0, otherwise. The normal distribution parameters are set experimentally, for  $R_t$ ;  $\mu_t = 6\text{m}$ ,  $\sigma_t = 15\text{m}$ , for  $R_p$ ;  $\mu_p = 7.5\text{m}$ ,  $\sigma_p = 15\text{m}$ . We set  $\mu_t$  smaller than  $\mu_p$  because



Wrong localization

Correct localization

Figure 3. The distance images used for calculating the verification score at *Suburban 1* image sequence. On top the tracks are shown with green (inlier) and red (outliers) in the distance image to the road network. In the bottom the road network (green) is shown in the distance image to classified road pixels in the image.

for  $d_{m_t}$  outliers are considered. The likelihood for a mean distance value is  $L_d = f(d_m)$ . The final score  $S$  that the image sequence  $I$  matches the roads  $R$  is the joint likelihood  $S = L_t L_p = f(d_t) f(d_p)$ ,  $d_t$  and  $d_p$  are handled as independent. At a correct match both likelihoods  $L_t$  and  $L_p$  have a high value, thus the final score is also high. For a false match at least one of the likelihoods has to decrease, thus the final score also decreases.

## 4. Experimental results

### 4.1. Synthetic data

The purpose of this test was to investigate whether the road map tiles (scenes) and the subsets of the roads in these tiles are unique or ambiguous over a city. We analyzed if the proposed *polyline based geometric hashing* method can distinguish the road network fractions over a city area, thus enabling geolocation recognition. Since the tracks are considered as subsets of the street network, we can generate synthetic tracks with known ground truth from the street map data.

#### 4.1.1 Test steps

- *Database creation:* We generate the hash entries for the search area  $A_s$  with a specific grid size as described in 3.2.1. Except the road nodes are not interpolated to limit the segment length to  $d_{max}$  since they are virtually identical to the vertices of the tracks.
- *Synthetic track generation:* In the search area  $A_s$  an axis aligned square (scene)  $A_r$  is extracted with side length  $d_s$ . In this  $A_r$  area either all streets are used or they are sampled. When sampling roads and line segments are randomly chosen with a ratio parameter defining how many roads  $r_r$  and segments  $r_s$  are kept. The remaining vertex points are randomly shifted in the line segment. The tracks are transformed by one random 2D rigid transformation  $T_r$  to a new area  $A_t$ . Since we shift the vertex positions only in the line segments (no noise is added, but the quantization of the geometric hashing can be considered as noise), a perfect location recognition is possible. The question is the ambiguity of the recognition.
- *Geolocation recognition:* A recognition as in 3.2.1 defines a 2D rigid transformation  $T_b$ . If the  $T_b$  transforms the corners of  $A_t$  back to their original location  $A_r$  the recognition is correct. The geolocation recognition is considered true only if it is correct, it has the highest number of votes, and no wrong recognition has the same number of votes – thus it is unambiguous.

### 4.1.2 Test configurations and results

We selected two search areas, one in New York City, US with an area of  $10.1 \times 11.1 \text{ km}^2$  containing 1495 km of streets. The second in Munich, Germany, with an area of  $14.1 \times 10.6 \text{ km}^2$  and 1560.4 km road length. See the map of the areas<sup>4</sup> on figure 4.

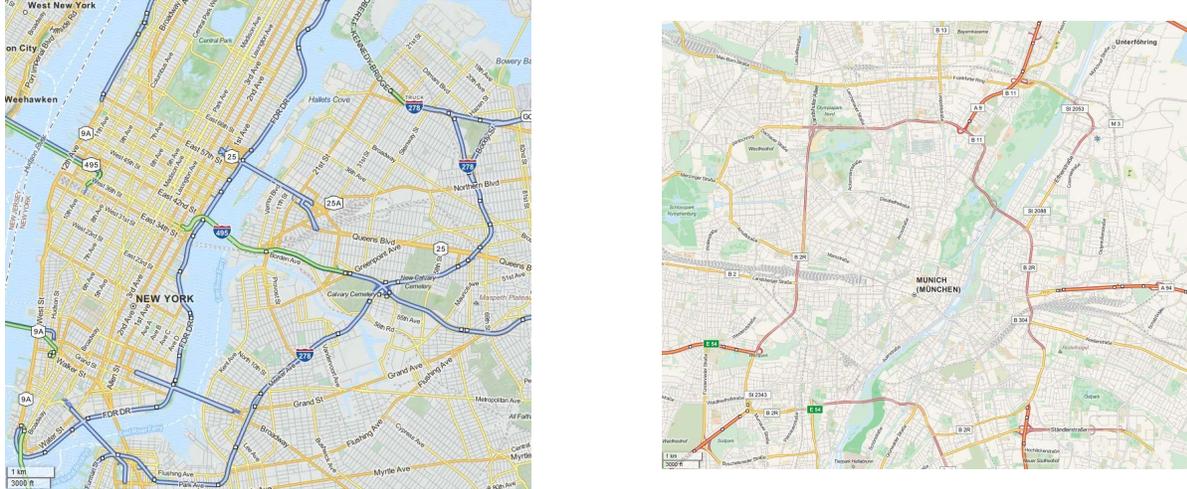


Figure 4. The search areas in New York City (left) and Munich from OpenStreetMap for the synthetic test.

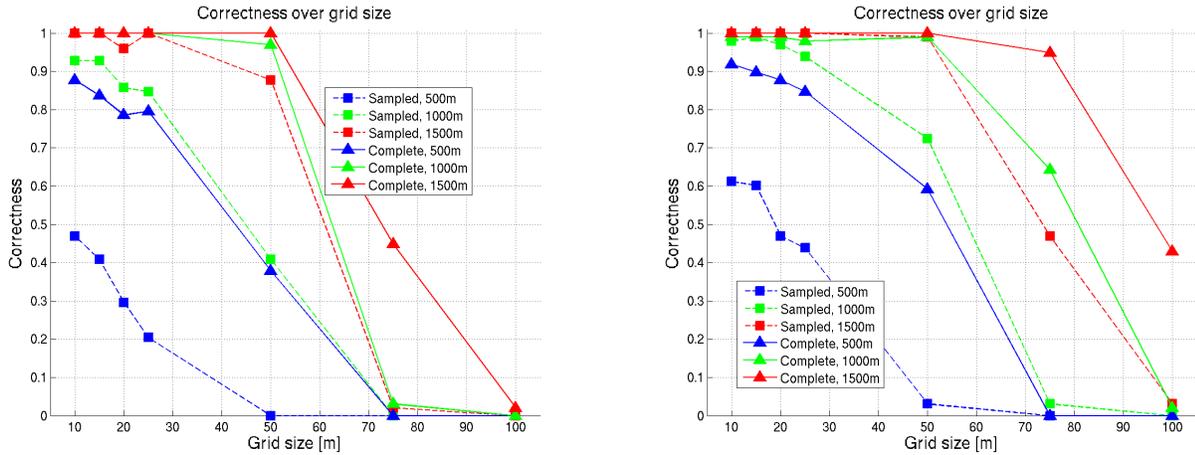


Figure 5. Correctness over grid size with different scene sizes and complete/sampled road network.

Grid Size [m]	10	15	20	25	50	75	100
New York	502.1	323.6	234.4	181.7	79.9	47.4	30.5
Munich	622.0	401.8	292.5	228.4	105.1	64.8	44.2

Table 1. Number of hash entries  $[10^6]$  in the database over the grid size.

The tiles of the road database are  $2500 \text{ m} \times 2500 \text{ m}$  with 50% overlap. We tested multiple grid sizes; see the table 1 for the values. 100 center points were generated randomly. Around each point scenes were generated with side lengths  $d_s = 500, 1000, 1500 \text{ m}$ . We analyzed the complete scene scenario and the more realistic case by sampling the roads with rate  $r_r = 0.5$ , the line segments with  $r_s = 0.3$  and shifting the node along the line segment with maximal 50 m.

<sup>4</sup>source: <http://www.openstreetmap.org>

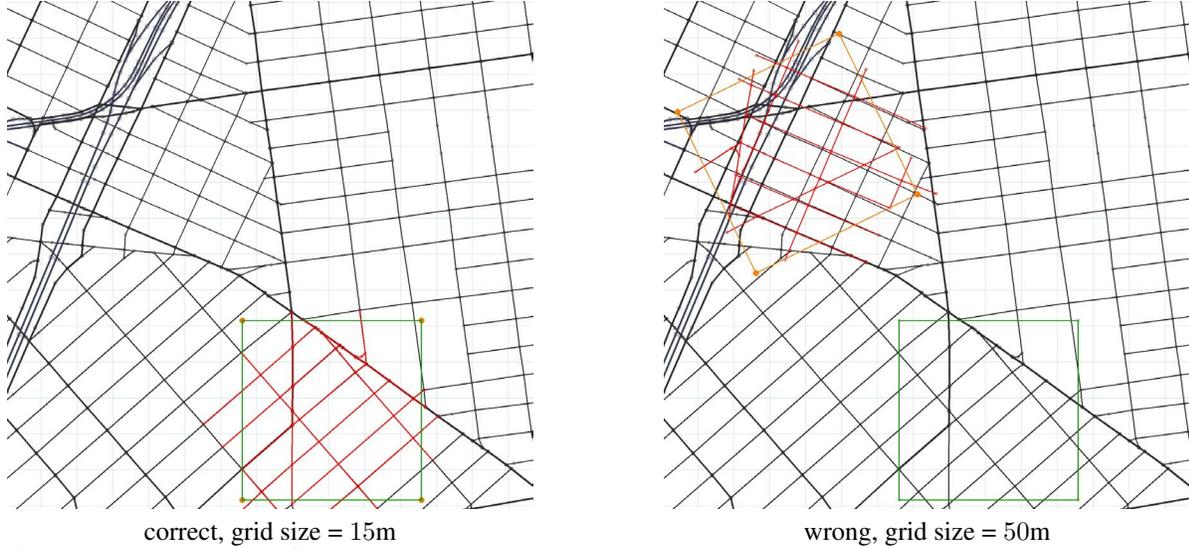


Figure 6. Example where the large  $d_g$  grid size leads to false geolocation. The green rectangle is the ground truth, the orange is the recognized geolocation. The road network is in black, the synthetic tracks are in red. The grid in the image is 100 m. The tracks exceed the scene to preserve the basis points.

We computed the  $correctness = T/(T + F)$  where  $T$  is the number of true and  $F$  is the number of false geolocalizations. On figure 5 the correctness was plotted over the grid size for the two cities with different sampling and different scene sizes  $d_s \times d_s$ .

Since the grid size of the geometric hashing both defines the number of bins and the rasterization of the line segments, it has an impact on the number of hash entries in the database. The number of hash entries over different grid sizes are in table 1. More hash entries imply higher memory demand and longer database retrieval and creation time.

The correctness values derived from our test indicate, as shown in figure 5, that the road network of a larger area is unique, though it might be ambiguous on local scale. The problematic areas occur where the whole street network is very regular like a chessboard or the scene contains a few straight roads. The unambiguity originates not simply from the perpendicular Manhattan World road network, typical in New York, but also from the identical distances between the streets. On a larger area it is more likely that a unique road will appear, which makes the whole road network also unique. The Figure 6 shows an example where the coarser grid size leads to wrong localization.

By increasing the scene size (i.e. flying longer over roads) a correct geolocation becomes possible based only on the road map. With a grid size of 15 m and scene size 1000 m  $\times$  1000 m we get a correctness  $> 0.9$  for all the scenarios and cities we have tested.

## 4.2. Real data – Aerial image sequences

We have tested the method on real aerial photos over Germany. 20 image sequences were acquired by a 21 MPixel Canon DSLR camera with a fixed focal length mounted in nadir direction on an airplane. The internal camera parameters were known. A GPS device registered the position of each image for the ground truth. Each image sequence contains 17 – 20 consecutive images taken with approximately 1 Hz frequency. The altitude over ground was between 1000 and 1500 m, which gives a GSD of 13 – 19 cm/pixel. The image sequences contain roads with car traffic. We grouped the scenes as urban (U), suburban (S), industrial (I) and motorway (M). This classification is not strict; we arbitrarily used it for this paper. The scenes were relatively flat, thus the earth surface could be modeled by a plane. Studying mountainous areas is planned for future work. The scenes *Suburban* 1 and 4, 5 and 2, 6 and 3 and *Urban* 3 and 4 are approximately over the same area but with two month difference. The original images were rectified with a 2 parameter radial distortion model. All the processing was done on these rectified images. The tiles of the road database were 2000 m  $\times$  2000 m with 50% overlap. The grid size  $d_g$  for the geometric hashing was 15 m. The 5 longest straight tracks in the scene were used for the shortlist retrieval. The size of the scenes was around 2000 m  $\times$  500 m.

### 4.2.1 Quantitative evaluation

The *PLBGHashing* delivers a shortlist of locations, while the verification gives a ranking of these locations. To decide if a geolocation was correct, the geolocalization was compared to the GPS values. If the GPS positions were correct, the mosaic image was inspected visually by overlaying it in Google Earth. We extracted a *TOPN* value, which indicates if there is a correct location in the first *N* list elements. The number of possible locations and orientations is very high even for a small search area. For a scene of 1000 m × 1000 m in a search area of 2 km × 2 km with 15 m translation steps, 5° angle steps, and 3 scales  $(\frac{1000}{15})^2 \times \frac{360}{5} \times 3 = 960000$ . A search area of 22500 km<sup>2</sup> consists of 22201 overlapped tiles, thus the total number of possibilities is in the 10<sup>10</sup> scale.

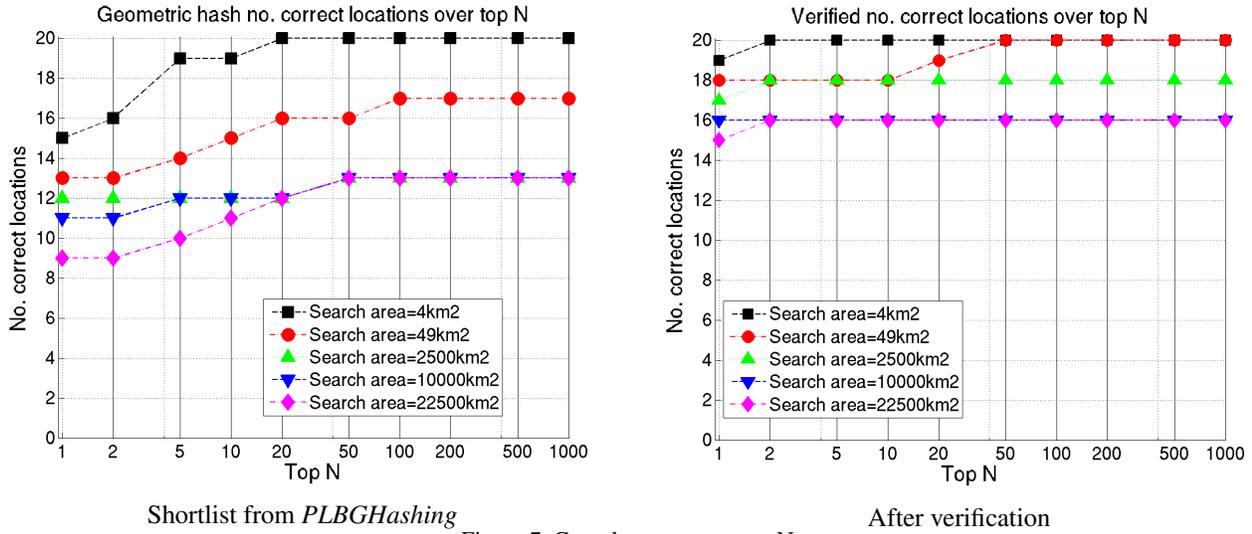


Figure 7. Completeness over top N.

In Figure 7 the aggregated number of correct localizations is plotted for different *N* values and search areas. There is a separate plot for the shortlist retrieval and the verification. The increase of Top1 by the verification was significant. We increased the search areas around the GPS position of the camera in steps 4, 9, 25, 49, 100, 225, 400, 625, 1225, 2500, 5625, 10000, 15625, 22500 km<sup>2</sup>. The Figure 8 shows for each scene the maximal size of the search area in which it was correctly located in the topN after the verification.

19 scenes out of 20 could be located correctly as top1. The scenes *Industrial 1, 2* contain tracks acquired in large parking lots (the tracking algorithm might extract wrong tracks on dense parked cars) outside the road network. If the number of outliers becomes greater than what the verification accepts, then the verification ranks the locations wrong. Thus even if the correct location was in the shortlist it can not be found. Therefore bad localization. The location of *I1* was wrong already at the smallest search area, while the *I2* could be localized only on 4 km<sup>2</sup>. *I2* contains more tracks over streets as *I1*, which has tracks only about one motorway and an industrial area, see the Figure 11. The other scenes show that if we have enough tracks on the roads, then the localization only weakly depends on the search area. On the Figure 10 and 11 we show the geolocalizations by projecting the mosaic images into the street network and as an overlay in Google Earth (the visualization as an image overlay is geometrically not fully accurate).

Some areas might be inherently ambiguous (e.g. chessboard like streets). An indicator for this ambiguity is if there are many candidates with the same number of votes in the shortlist and in the list after validation.

We defined the grid size for the geometric hashing as 15 m. This makes the method robust to small displacements in the tracks and it can handle multi-lane roads. If we assume 3 m wide lanes, then 5 lanes are quantized to the same grid (i.e. the localization is invariant to this displacement). Roads with more than 5 lanes are sparse as large roads (e.g. motorways) in OSM are stored as two separate roads for each direction.

Our method is agnostic to splits or merges of the tracks as the assignment between line segments and tracks is not relevant. A track consisting of 10 line segments has the same effect as if it is split to 10 individual tracks.

The search area 150 km × 150 km around the *Urban* scenes contain nearly 32000 km of roads, compiling to 2262M entries in the hash which requires 30500 MB memory. For the *Urban 1* scenes containing many tracks the retrieval took 9 minutes on the largest search area, for a scene with less tracks the retrieval is faster, for the *Suburban 1* it took 2 minutes. The

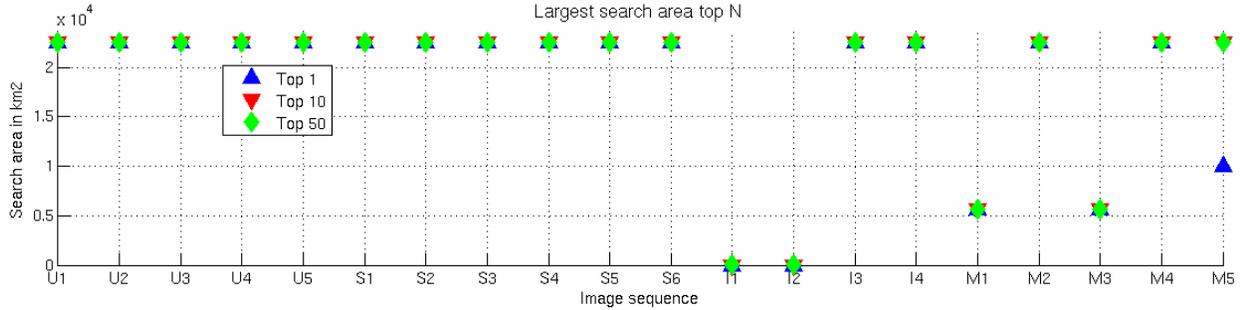


Figure 8. The largest search area for each image sequence where the correct verified geolocation was in the TOPN in  $\text{km}^2$

average accuracy of the geolocalization is around 25 m, this also depends on the accuracy of the road data (A more accurate location could be acquired by optimizing the resulting location, but this was not our goal in this paper). Our implementation was written in C++, multi threaded. The experiments run on an Intel Xeon E5-1650v2 processor with 6 cores. The verification took around 36 ms per shortlist element. Since the geometric hashing is very suitable for parallel implementation, the retrieval time on large search areas can be significantly reduced by utilizing more cores or machines.

#### 4.2.2 Comparison to simple chamfer matching

We consider chamfer matching [5] with brute force search as a baseline. Without the *PLBGHashing* for shortlist generation, by only a brute force search and verification of each position the number of required verifications would be very high. Since the processing time of the verification is relatively long (36 ms per location), it would take  $960000 \times 0.036s = 9.6$  hours on  $2 \text{ km} \times 2 \text{ km}$  search area  $10^{10} \times 0.036s = 3.6 \times 10^8s = 11$  years for the largest search area (on a single thread). This is too long for practical usage.

#### 4.2.3 Comparison to graph matching

To compare our method to graph matching, we reimplemented the method in [16]. We formulated this bipartite graph matching problem with many-to-one matches as one of inference in a Markov random field (MRF). The variables (nodes) are the segments in the tracks, while the states are the segments in the map plus a state defining no match. We define pairwise potentials between all variables. The pairwise potential consists of the sum of differences between 3 binary features (i.e. the relative orientation between lines, the minimum distance between the line endpoints and the distance between the segment center points). The energy function of this MRF is similar as the (2) energy function of [16]. The number of variables is  $T$ , the number of pairwise connections is  $T(T - 1)/2$  where  $T$  is the number of track segments. Each variable has  $M + 1$  states (where  $M$  is the number of map segments), thus the pairwise potential is a matrix of  $M + 1 \times M + 1$  values. The MAP inference in this MRF gives the matching between the tracks and the maps. The inference in this MRF is an NP-hard problem as it contains many loops and the pairwise potential can not be guaranteed to be regular. Therefore we have to perform approximate inference. We refer the reader to paper of Kolmogorov and Zabini [13] for the conditions of an MRF inference problem being NP-hard. We applied the AStar inference algorithm [6] of the OpenGM software [2].

We have one difference to the method in [16]. [16] has an approximation for the initial direction of the image and therefore they define possible matches between line segments only to those within a given orientation difference. This can reduce the number of states per variable significantly. For our problem this is not the case, since we have no information about the orientation of our images. We have to match all track segments to all possible map segments.

We used synthetic tracks (the segments of the map randomly transformed) on a limited search area (max  $700 \times 700 \text{ m}^2$ ) to validate if our implementation can provide correct localizations. As long as both the number of map segments (labels) and the track segments (variables) were kept low ( $< 200$ ) the approximate MAP inference provided a decent segment assignment and thus correct localization. But if we moved to larger search areas (e.g.  $1000 \times 1000 \text{ m}^2$ ) or real tracks which consist of more than 200 segments and has outliers, the inference could not find a solution in a reasonable time (30 minutes) and the memory need for the algorithm became also too high for practical use. We show an example for this on Figure 9.

The increase of  $T$  and  $M$  leads to a difficult inference problem and the memory consumption becomes also a problem, since the number of values stored for pairwise features is  $(M + 1)^2 T(T - 1)/2$ . If  $T = 200$  and  $M = 1000$  we already need

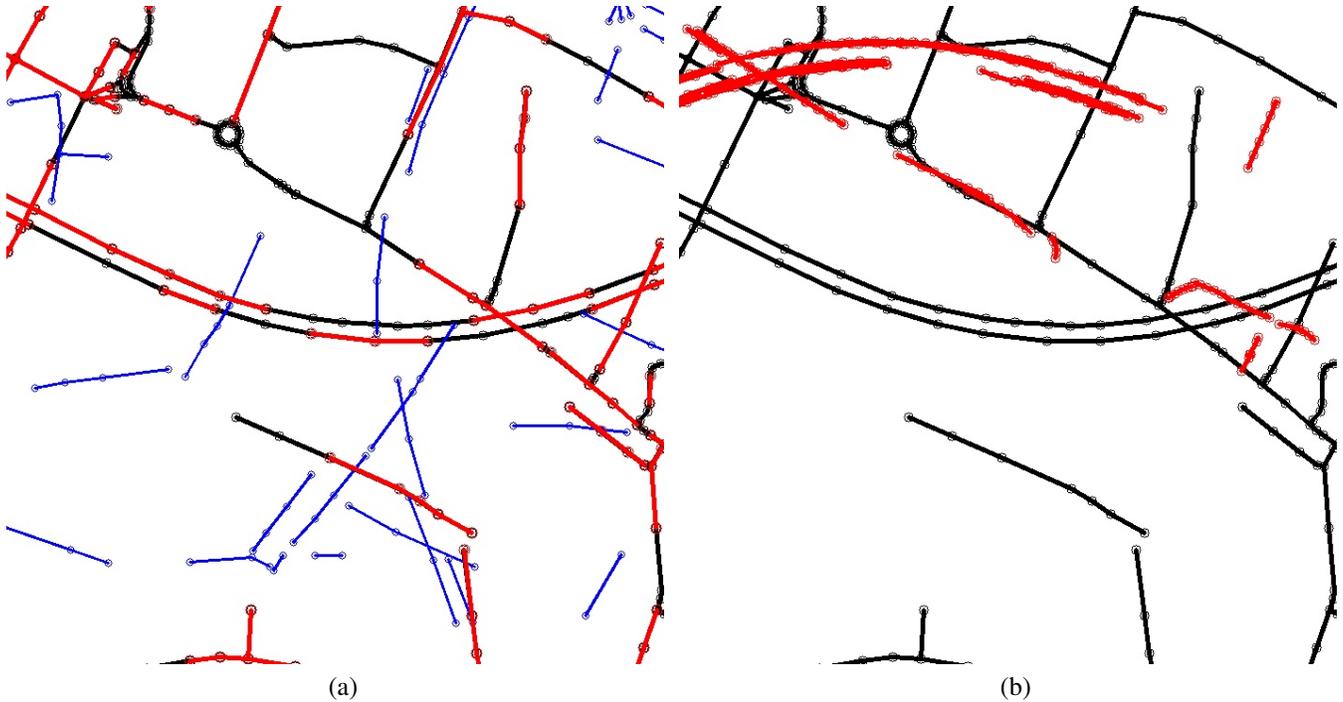


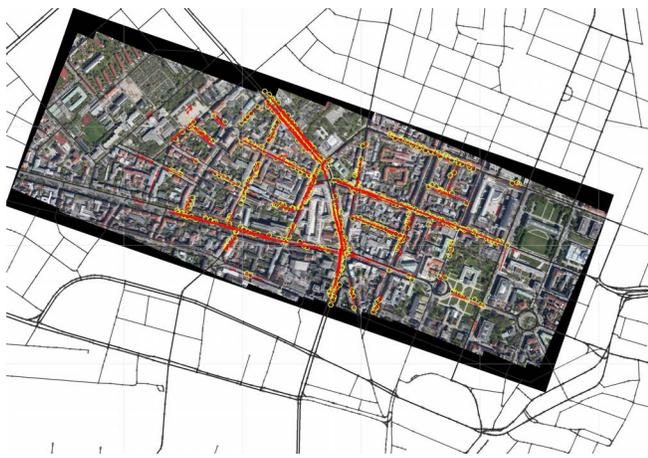
Figure 9. Graph matching based localization [16] on the *Suburban 1* sequence using  $700 \times 700m^2$  search area. The (a) shows the case with synthetic tracks (random sampling of the road network). Here we have 110 variables each with 174 states (the number of map segments). The inference gave an assignment between the segments which define a correct localization. The map segments are shown in black, the sampled road segments transformed in blue and the located segments in red. On (b) we use real tracks defining 353 variables. The inference could not find a matching defining a correct geolocation in reasonable time (30 minutes). The localized tracks after 30 minutes inference running time are shown in red.

to store  $\approx 2 \times 10^{10}$  values.

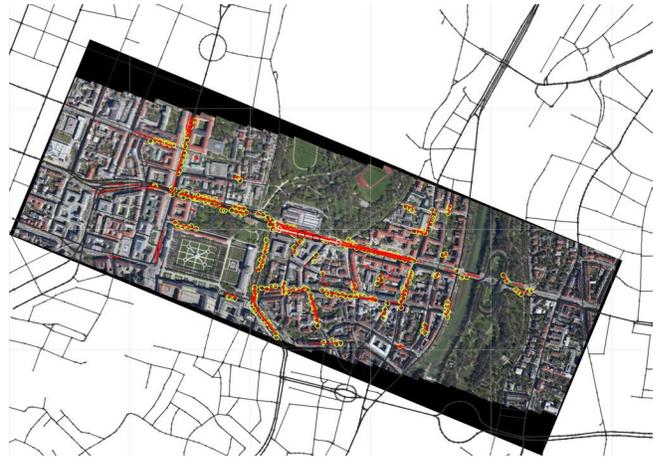
Graph matching as in [16] provides an energy formulation giving a matching invariant to rotation and translation. However the minimization of this energy function is intractable for large scale problems with many variables and states. This prevents the practical application for large scale geolocation.

## 5. Conclusion and outlook

In this paper we have addressed the problem of finding the geolocation of an aerial image sequence over a larger area. Our method utilizes only the visual information and the road map which is easily accessible. We have demonstrated the effectiveness of our method in tests with synthetic data derived from the road network and real photos captured over various scenes (urban, suburban, industrial, motorway). In practice much longer image sequences could be used covering more roads, thus making the localization easier. The limitation is the presence of road traffic at the area and the flat ground assumption. In our future work we plan to address these limitations, e.g. by applying a road detector the method could localize scenes without traffic and also single images, a 3D model of the scene generated by dense stereo could extend the method to areas with relief.



(a) *Urban 1.*



(b) *Urban 2.*



(c) *Urban 3.*



(d) *Suburban 1.*



(e) *Suburban 5 (90° rotated).*



*Suburban 5 an an overlay in Google Earth*

Figure 10. Geolocalization results. The mosaic images are projected into the street network. The roads are in black, the tracks are highlighted with red lines, the locations of votes from the *PLBGHashing* are marked with yellow circles. (When an image scene is only partially overlapped with a model during the shortlist retrieval, there are no vote circles at every matching track-road locations (e.g. on *Urban 1-3*)) On (f) the geolocalized mosaic image with the tracks is overlaid in Google Earth. All these scenes were correctly located on a 22500 km<sup>2</sup> search area.



(a) *Industrial 1* false location.



(b) *Industrial 3*. (90° rotated)



(c) *Motorway 2*.



(d) *Motorway 2* as an overlay in Google Earth.



(e) *Motorway 3*.



(f) *Motorway 5* (90° rotated).

Figure 11. Geolocalization results. The mosaic images are projected into the street network. The roads are in black, the tracks are highlighted with red lines, the locations of votes from the *PLBGHashing* are marked with yellow circles. On (d) the geolocalized mosaic image with the tracks is overlaid in Google Earth. The (a) shows the scene *Industrial 1* located false on a 4 km<sup>2</sup> search area. The causes for this are; the scene contains 2 roads which makes it inherently ambiguous, multiple tracks are extracted on a parking lot, outside of the road network. Since there are more tracks in the parking lot than on the road, the verification matching handles the correct tracks as outliers. The color based road detection also does not work properly, since a parking lot has similar color as the road. The error is approximately 100m.

## 6. Acknowledgement

This work was funded by the *Vabene++*<sup>5</sup> project of the German Aerospace Center (DLR).

## References

- [1] Saad Ali and Mubarak Shah. Cocoa - tracking in aerial imagery. In *Proc. Int. Conf. on Computer Vision*, 2005. 3
- [2] B. Andres, Beier T., and J. H. Kappes. OpenGM: A C++ library for discrete graphical models. *ArXiv e-prints*, 2012. 11
- [3] Georges Baatz, Kevin Köser, David Chen, Radek Grzeszczuk, and Marc Pollefeys. Leveraging 3d city models for rotation invariant place-of-interest recognition. *Int. J. Comput. Vision*, 96(3):315–334, February 2012. 2
- [4] Georges Baatz, Olivier Saurer, Kevin Köser, and Marc Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *Computer Vision ECCV 2012*, Lecture Notes in Computer Science, pages 517–530. Springer, 2012. 3
- [5] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: two new techniques for image matching. In *Proceedings of the 5th international joint conference on Artificial intelligence - Volume 2, IJCAI'77*, pages 659–663, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc. 6, 11
- [6] Martin Bergtholdt, Jörg Kappes, Stefan Schmidt, and Christoph Schnörr. A study of parts-based object class detection using complete graphs. *International Journal of Computer Vision*, 87(1):93–117, 2009. 11
- [7] Gunilla Borgefors. Distance transformations in digital images. *Comput. Vision Graph. Image Process.*, 34(3):344–371, June 1986. 6
- [8] Marcus A. Brubaker, Andreas Geiger, and Raquel Urtasun. Lost! leveraging the crowd for probabilistic visual self-localization. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3057–3064, 2013. 3
- [9] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998. 4
- [10] Patrick Gros, Olivier Bournez, and Edmond Boyer. Using local planar geometric invariants to match and model images of line segments. *Computer Vision and Image Understanding*, 69(2):135 – 155, 1998. 3
- [11] Achim Hornbostel, Manuel Cuntz, Andriy Konovaltsev, Gtz C. Kappen, Christian Httich, Carlos A. Mendes da Costa, and Michael Meurer. Detection and suppression of ppd-jammers and spoofers with a gnss multi-antenna receiver: Experimental analysis. In *European GNSS Conference*, 2013. 1
- [12] S. Kluckner, G. Pacher, H. Grabner, H. Bischof, and J. Bauer. A 3d teacher for car detection in aerial images. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007. 3
- [13] V. Kolmogorov and R. Zabini. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, Feb 2004. 11
- [14] K. Kozempel and R. Reulke. Camera orientation based on matching road networks. In *Image and Vision Computing New Zealand, 2009. IVCNZ '09. 24th International Conference*, pages 237–242, 2009. 3
- [15] K. Kraus, I. Harley, and S. Kyle. *Photogrammetry: Geometry from Images and Laser Scans*. Number Bd. 1 in De Gruyter textbook. Bod Third Party Titles, 2007. 2
- [16] StanZ. Li, Josef Kittler, and Maria Petrou. Matching and recognition of road networks from aerial images. In G. Sandini, editor, *Computer Vision ECCV'92*, volume 588 of *Lecture Notes in Computer Science*, pages 857–861. Springer Berlin Heidelberg, 1992. 3, 11, 12
- [17] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision ECCV 2012*, volume 7572 of *Lecture Notes in Computer Science*, pages 15–29. Springer Berlin Heidelberg, 2012. 2
- [18] Tsung-Yi Lin, Serge Belongie, and James Hays. Cross-view image geolocation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 891–898, 2013. 3
- [19] Yuping Lin and G. Medioni. Map-enhanced uav image sequence registration and synchronization of multiple image sequences. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–7, 2007. 2
- [20] Volodymyr Mnih and Geoffrey E Hinton. Learning to label aerial images from noisy data. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 567–574, 2012. 3
- [21] Rupert Müller, Thomas Krauß, Mathias Schneider, and Peter Reinartz. Automated Georeferencing of Optical Satellite Data with Integrated Sensor Model Improvement. *Photogrammetric Engineering and Remote Sensing*, 71(1):61–74, 1 2012. 2

---

<sup>5</sup><http://www.dlr.de/vabene/>

- [22] Tomohiro Nakai, Koichi Kise, and Masakazu Iwamura. Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. In *In Lecture Notes in Computer Science (7th International Workshop DAS2006)*, pages 541–552. Springer, 2006. 4
- [23] F. Stein and G. Medioni. Efficient two dimensional object recognition. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume i, pages 13–17 vol.1, 1990. 4
- [24] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, May 2004. 3, 4
- [25] Caixia Wang, A Stefanidis, and P. Agouris. Relaxation matching for georegistration of aerial and satellite imagery. In *Image Processing, 2007. IICIP 2007. IEEE International Conference on*, volume 5, pages V – 449–V – 452, Sept 2007. 3
- [26] Jan D. Wegner, Javier A. Montoya-Zegarra, and Konrad Schindler. A higher-order crf model for road network extraction. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1698–1705, 2013. 3
- [27] R Wilson and E R Hancock. Relaxation matching of road networks in aerial images using topological constraints. *Sensor Fusion*, 2059:444–455, 1993. 3
- [28] H.J. Wolfson and I. Rigoutsos. Geometric hashing: an overview. *Computational Science Engineering, IEEE*, 4(4):10–21, 1997. 4
- [29] Changchang Wu. Towards linear-time incremental structure from motion. In *3DTV-Conference, 2013 International Conference on*, pages 127–134, 2013. 4
- [30] Changchang Wu, Friedrich Fraundorfer, Jan-Michael Frahm, Jack Snoeyink, and Marc Pollefeys. Image localization in satellite imagery with feature-based indexing. In *Commission III, ISPRS Congress 2008 Beijing*, volume XXXVII, pages 197–202, 2008. 2
- [31] Jiangjian Xiao, Hui Cheng, H. Sawhney, and Feng Han. Vehicle detection and tracking in wide field-of-view aerial video. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 679–684, June 2010. 3
- [32] Yan-Tao Zheng, Ming Zhao, Yang Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the world: Building a web-scale landmark recognition engine. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1085–1092, 2009. 2