2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

# LASOM: Location Aware Self-Organizing Map for Discovering Similar and Unique Visual Features of Geographical Locations

Dmitry Kit Yu Kong Yun Fu

***Abstract*— Can a machine tell us if an image was taken in Beijing or New York? Automated identification of the geographical coordinates based on image content is of particular importance to data mining systems, because geolocation provides a large source of context for other useful features of an image. However, successful localization of unannotated images requires a large collection of images that cover all possible locations. Brute-force searches over the entire databases are costly in terms of computation and storage requirements, and achieve limited results. Knowing what visual features make a particular location unique or similar to other locations can be used for choosing a better match between spatially distance locations. However, doing this at global scales is a challenging problem. In this paper we propose an on-line, unsupervised, clustering algorithm called Location Aware Self-Organizing Map (LASOM), for learning the similarity graph between different regions. The goal of LASOM is to select key features in specific locations so as to increase the accuracy in geotagging untagged images, while also reducing computational and storage requirements. Different from other Self-Organizing Map algorithms, LASOM provides the means to learn a conditional distribution of visual features, conditioned on geospatial coordinates. We demonstrate that the generated map not only preserves important visual information, but provides additional context in the form of visual similarity relationships between different geographical areas. We show how this information can be used to improve geotagging results when using large databases.**

## I. Introduction

Understanding where user generated images come from (e.g. Beijing or New York) can be a great source of contextual information. For example, object detection can be constrained to search for location-specific objects.

The number of photos taken at a location can provide information about the popularity of a particular place, which can be used in tourist recommendation systems [1]. Image search and retrieval can also benefit greatly from this work [2]. Finding videos and images related to a particular location or even places mentioned in a news story is of great interest to mass media [3]. Images and videos recorded in the same area tend to be related in terms of activity and content. Finally, determining where an image was taken is valuable to the intelligence community for use in surveillance.

The availability of geotagged images on sites such as Flickr has allowed researchers to explore the problem of automatic geotagging of images and videos that are missing such information. Most successful approaches have combined media tags with global image descriptors and authorship information [4]. Yet, there is plenty of untapped information that still exists in image content. Understanding how features are distributed in the world and using such information can improve geotagging performance. Specifically, this information can be used for building a list of geographical regions where a query might have originated.

Dmitry Kit, Yu Kong and Yun Fu are with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA (email: {d.kit, yukong, yunfu}@ece.neu.edu).

This research is supported in part by the Office of Naval Research award N00014-12-1-1028, Air Force Office of Scientific Research award FA9550-12-1-0201, and IC Postdoc Program Grant 2011-11071400006

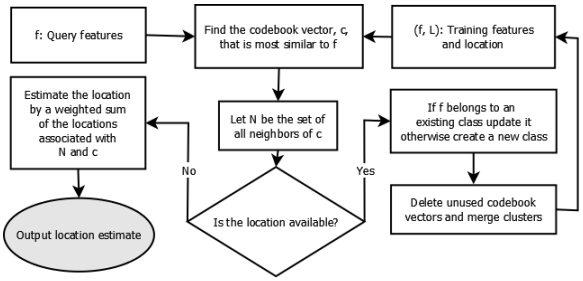



Fig. 1: Overview of the LASOM algorithm, which uses a dictionary of codebook vectors to determine the location of a query image.

Due to a huge variability of visual features across the world, geolocation is inherently a big data problem. Since hand labeling all training images is unfeasible, employing unsupervised methods would be much more effective. One solution is to assign the location of the first best match to a query image [5]. This method can perform fairly well, but requires an extremely large database of images, increasing resource requirements and amplifying noise.

Alternatively, image similarity graphs [6] can be used to reduce the search space to a small set of most relevant images. However, considering the size of the databases, capturing similarity relationship between all images is difficult and can increase the computation requirements for adding new images to the database. Furthermore, manually sifting through such databases becomes impractical.

Many geolocation techniques will first perform some form of clustering before training a classifier to detect which cluster a query image belongs to. These clusters are either done on the location information [7], [8], [9] or on the visual features [6], using such methods as k-means or mean-shift. In this paper we propose a new on-line unsupervised clustering algorithm, called Location Aware Self-Organizing Map (LASOM), that can be used for estimating the density distribution of one variable conditioned on another. The algorithm, outlined in Figure 1, explicitly uses location information to determine whether we need to store a training

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

image's feature[1] at a particular location or if the information contained in a training image is redundant. Self-Organizing Map (SOM) is then used at that location to learn the signature of that region by modifying local representation to capture the novelties of new training samples. The noise reduction step removes outliers and merges similar clusters to further reduce computational and storage requirements.

SOM [10] algorithms are designed to incrementally build relationship graphs without supervision. However, current SOM algorithms are not suitable for solving the general geolocation problem and suffer from two major issues. First, both the visual features and the locations are highly variable. Second, a large amount of training data is required to demonstrate this variability. LASOM separates the task of clustering the visual features from spatial clustering, making it intuitive for the geolocation problem. It also requires fewer complex parameters than other algorithms.

This work provides an important contribution in the form of a new method that reduces noise in a dataset as well as provides compression, enabling it to scale to very large databases. This compression does not reduce the performance of location estimates and moreover, it can be improved using the learned relationships between images. These contributions bring us closer to solving the geolocation problems which span large spatial distances and require huge databases. Furthermore, using on-line algorithms allows adding new data without having to restart the learning process. To the best of our knowledge, this work is the first attempt at applying unsupervised on-line clustering algorithms to the geolocalization problem.

## II. Background

The present work is a fusion of two different areas of research. The interest in the geolocation problem has made it an active research topic with recent work showing that graph-based representations can aid greatly in the task. Unsupervised incremental clustering methods are a class of algorithms that can be used to construct such data structures and improve geolocation performance. We will first review the efforts to solve the geolocation problem, before discussing a class of unsupervised clustering algorithms called Self-Organizing Maps.

### A. Geolocation

At its core, the geolocation problem is about assigning coordinates to an unannotated image or a video [8]. Hays and Efros [5] showed that a simple scene matching approach can achieve respectable performance. This required a large database of images and comparing each query image against every item in the database. Since the goal of our work is similar, we will use k-nearest neighbor search (k-NN) as a baseline to show that our approach scales and provides better performance through noise reduction and by learning the relationship between image features and locations.

[1]In this paper we learn the distributions of different feature types (e.g. color histogram and texture) separately.

Support Vector Machines (SVM) have been used with some success to discriminate between different regions [9], [11], [6]. The use of SVMs, however, requires a set of positive and negative examples to be generated. It is not always clear how to do this. Crandall et al. [11], for example, first clustered their dataset and then selected the top ten clusters as class labels. This would not work for locating an image that comes from a low density cluster. Furthermore, k-NN often works very well at geotagging popular locations [5], despite the claim made in [11] that image features are a poor choice for localizing images in global databases.

Other approaches have augmented the image or video information with external sources such as authorship information [4], time between photos made by a single person [12], textual tags [13], [14], [4], [15]. In this work we assume that none of this information is available for a query image and that the image has to be classified based only on visual features. For a more detailed discussion of the geolocation problem, refer to [16].

### B. Self-Organizing Maps

Cao et al. [6] have shown how similarity relationships in a form of a graph can be used for geolocating images. Their graph structure did not capture geotags associated with images and was only used for retrieving visually similar images based on a query image. The results were used to reduce the time it took for a more complex image matching method (geometric verification). In contrast, our goal is to create graphs that explicitly represent geospacial information as well as visual features.

We also draw inspiration from our previous work [7], where we proposed using Growing Neural Gas (GNG) [17], a variant of SOM algorithm, for constructing visuo-spatial representations of small environments. They achieved considerable compression, representing over 25,000 input images using only 500 codebook vectors, while still maintaining a representation that was detailed enough for identifying location specific information. Shao et al. [18] proposed a method for improving the GNG algorithm called Enhanced Self-Organizing Incremental Network (ESOINN). They changed the way new nodes were added to the graph, how the connections were made, and how to determine outliers.

From our experimentation on a large dataset we found that neither GNG nor ESOINN could learn a representation that produces accurate geotagging performance. In these approaches the location is not an explicit concept and the relationship between a visual feature and a geotag must be captured implicitly. This requires a large amount of training data, which requires longer training times. Using high dimensional data with ESOINN resulted in the creation of many clusters that were not connected to any other cluster, causing most of the training information to be lost at the noise reduction step. By making location an explicit concept, we provide a method for specifying what relevant information should be stored and when.

## III. METHODOLOGY

The goal of automatic geotagging is to find a list of probabilities for each possible location, $L$, given a query feature vector, $f$. More specifically we want to find $\mathrm{argmax}_L\, p(L|f)$. Calculating this directly is very difficult, due to the large variability of visual data at each possible location. We can, however, approximate the solution by $\mathrm{argmin}_{f_l} \parallel f - f_l \parallel$ [5], where $f_l$ is an image feature vector at location $L$. In other words, the best match in a very large database should correspond to the most likely location of the query feature. Such an approach suffers from two issues. First, the number of comparisons that must be made grows linearly with the size of the database, $O(|f_l|)$. Second, this method suffers from noise and increasing the size of the database only aggravates the problem. Since no other contextual information is available, obtaining high performance using this method has been difficult. In the next subsection we discuss the details of a new on-line, unsupervised clustering algorithm that compresses large databases by only keeping the information required to identify a specific location.

### A. Location Aware Self-Organizing Map

LASOM is a specialized algorithm for learning the distribution of one feature conditioned on another one. Spatial constraints will be used to learn the distribution of visual features at particular geographical locations. The algorithm is based on a class of algorithms known as Self-Organizing Maps (SOM) [10]. Our approach to the geolocation problem requires us to build visual signatures for geospatial areas of particular size. The region size in this case acts as a constraint on the location variable, while visual features stored at that location represent the visual signature of that location. During training LASOM estimates a solution for this problem using two key steps.

In step one, LASOM uses the explicit knowledge of space to assign an input feature $f$ to a codebook vector, $c_1$ if and only if the visual features of $c_1$, $W_{c_1}$ is the closest vector in feature space $\parallel W_{c_1} - f \parallel$ and is also within the geospatial constraint (i.e., within a distance threshold). This step specifically targets ambiguities between different spatial regions. A blue house in one town may look similar to a blue house in another town. Traditionally, an SOM would average the distance between the two locations, as that would minimize the average error. In our case this is not the desired solution, since we want to record information about both locations. If $c_1$ is geospatially distant then LASOM will consider assigning $f$ a neighbor of $c_1$ if they are within $f$'s region. Since neighborhoods are formed based on visual similarity, neighbors of $c_1$ are also similar to $f$. Finally, if nothing is found, $f$ is added to the set of codebook vectors and an edge between $c_1$ and $f$ is added to the set of edges. This allows future inputs to resolve ambiguities between different regions. How LASOM determines class membership and the need to add a class is one of its novelties.

In the second step LASOM uses a novel method for reducing the codebook size. Instead of removing unlikely codebook vectors LASOM attempts to merge them with other codebook vectors. By waiting to do this after a predetermined number of learning iterations, LASOM allows time for codebook vectors to be selected. Selected vectors are updated and move around geographically as well as in feature space. Perhaps a very similar codebook vector started outside some node's spatial constraints, but with time has shifted to reside within it. Consider a city street where all houses are painted green. Perhaps initial training steps introduced input vectors from the opposite ends of the street. Clearly, no assumption can be made about the point between these two inputs. However, with time, some training samples will land closer to one node and at other times samples will be closer to the other, shifting them both towards the center of the street. After a certain point it will become clear that both codebook vectors are being drawn to the same physical location. The next three sections will formalize these ideas.

### B. Training

The LASOM algorithm starts with a small codebook dictionary, which can be randomly generated or created from training samples. Given a query feature vector, $f$, LASOM finds the top two matching codebook vectors, $c_1$ and $c_2$ in the current dictionary, $D$ using some distance metric (e.g. $c_1 = \mathrm{argmin}_{c \in D} \parallel f - W_c \parallel$, where $W_c$ are the features associated with codebook vector $c$). The goal of LASOM is to minimize $\parallel f - W_c \parallel$ function. We use two distance functions depending on the feature type (e.g. histogram or vector). This will be discussed in Section 4.1. $c_1$ may or may not be close to the actual location of the feature vector, $L_f$ . Each codebook vector has GPS coordinates associated with it. During training, these coordinates are used to determine whether the input feature $f$ belongs to the codebook vector $c_1$ or not by limiting the maximum distance between them, $H(L_f, L_{c_1}) < \delta$. The 'haversine' distance formula was used to compute the distance, H(x,y), along the surface of the earth.

If the physical distance between $f$ and $c_1$ is not within this threshold, but there is a neighboring node that is very similar to the input and is within the distance threshold then $c_1$ becomes that neighbor and $c_2$ is replaced with the old $c_1$ . In this process, all the scores are transformed to a Gaussian, $\mathcal{N}(0, 1)$, by subtracting the mean and dividing by the standard deviation. A neighbor is considered similar if its transformed score is less than $-3$, or 3 deviations from the mean. If there is a $c_1$ within a threshold distance of $f$ then $c_1$ is updated to reduce $\parallel f - W_c \parallel$. If we had to replace the global match with its neighbor, this has the effect of making it more similar to the global match, $\Delta W_{c_1} = \epsilon \times (f - W_{c_1})$, where $\epsilon$ is a learning parameter. Intuitively this process can be seen as moving a codebook vector in feature space until the center, or average, of that space is found. This movement is constrained by neighboring nodes. The on-line nature of the algorithm gives greater weight to more recent observations than to old training samples. To prevent $c_1$ from *forgetting* old inputs $\epsilon$ is defined as $\epsilon(t) = \frac{1}{t}$. In LASOM $t$ is set to the number of times a particular codebook vector

**Algorithm 1:** The training algorithm for LASOM

**Result**: $G = (E, V)$, AgeMatrix: Staleness of an edge, WinningTimes: # of times $v \in V$ was a winning vector

**Data**: $(L_f,\ f)$: Set of Training Samples

**while** *there are more training samples* **do**
- $c_1 \leftarrow \text{argmin}_{c \in E} \parallel f_t - W_c \parallel$;
- $c_2 \leftarrow \text{argmin}_{c \in E \setminus c_1} \parallel f_t - W_c \parallel$;
- Let $N$ = Neighbors of $c_1$;
- **if** $H(L_{f_t}, L_{c_1}) \geq \delta$ *AND* $\exists_{n \in N} such\ that \parallel f_t - W_n \parallel$ *3 deviations closer to* $f_t$ *as any other node AND* $H(L_{f_t}, L_n) < \delta$ **then**
  - $c_2 \leftarrow c_1$;
  - $c_1 \leftarrow n$;
- **if** $H(L_{f_t}, L_{c_1}) < \delta$ **then**
  - $W_{c_1} = W_{c_1} + \epsilon(WinningTimes(c_1)) \times (f - W_i)$;
  - increment $WinningTimes(c_1)$ by 1;
  - If $(c_1, c_2) \notin E$ add an edge $(c_1, c_2)$ to $E$;
  - $AgeMatrix(c_1, c_2) = 0$;
  - $\forall_{n \in N \setminus c_2}$ increment $AgeMatrix(n, c_1)$ by 1;
- **else**
  - Let $C = (L_{f_t}, f_t)$;
  - Add the node $C$ to $V$;
  - Add an edge $(C, c_1)$ to $E$;
- **forall the** $(e_1, e_2) \in E$ **do**
  - **if** $AgeMatrix((e_1, e_2)) > \Gamma$ **then**
    - remove $(e_1, e_2)$ from $E$;
- **if** *current trial is a multiple of* $\lambda$ **then**
  - $G \leftarrow MergeClusters(G, AgeMatrix, WinningTimes)$;

won (similar to [18], [19]). After winning many times, it can be assumed that a vector is representative of that location. Finally, an edge is added between $c_1$ and $c_2$ to signify that these two codebook vectors represent similar information. Unlike other SOM algorithms neither $c_2$ nor any existing neighbors of $c_1$ are updated. This is because LASOM is designed to handle highly variable data and makes no assumptions about what neighboring areas might look like. The location, $L_{c_1}$ is updated along the surface of the earth based on bearing and distance to the input.

Finally, if the distance threshold is not satisfied (Equation 3.1), $f$ and $L_f$ become a new codebook vector. This new vector is connected to $c_1$, since it is visually similar to that codebook vector. This entire process can be seen as spatial binning of visual information, where the bin centers do not have to be determined a-priori, and are based on the visual variability in a particular region (i.e. more codebooks are dedicated to highly variable areas). The training procedure is summarized in Algorithm 2.

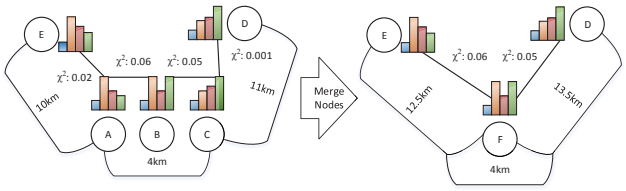



Fig. 2: Hypothetical example of merging multiple codebook vectors when the distance threshold is 4 km. The feature associated with the codebook vector B are closer to vectors A and C. The feature of vectors A and C are closer to E and D, respectively. Since the three nodes are within the threshold of 4 km, vectors A, B and C, are merged into a new codebook vector F.

### C. Noise reduction

The above procedure generates a great number of classes, many of which are created due to noise and others because two or more codebook vectors were not at their appropriate centers before a particular training sample was observed. LASOM has two mechanisms for dealing with this. First, a counter associated with all the edges between $c_1$ and its neighbors is incremented. For newly created edges this counter is initialized at zero. For existing edges, a Hebbian rule that resets the counter for the edge between $c_1$ and $c_2$ every time it is determined that $f$ belongs to $c_1$. An edge denotes that the two codebook vectors connected by it are visually similar. The counter specifies the freshness of this relationship. If the counter reaches a value, above a threshold, $\Gamma$, the edge is removed, signifying that the relationship is no longer valid. A codebook vector with no neighbors is removed. This is a form of local outliers detection. If one node keeps winning, but its neighbor never does (or is never a second best node), then more than likely that codebook vector is an outlier.

The previous rule is a local noise detection rule, as the counters are only incremented for $c_1$'s neighbors. The second mechanism is a global method for dealing with the ever changing topology of the network. As nodes get added, removed and updated, a particular spatial region may become over-saturated with similar codebook vectors. After a pre-specified number of rounds, $\lambda$, LASOM attempts to remove these redundancies. Each vector in $D$ becomes an input vector. The list of vectors, sorted by the feature distances to the input vector, is examined and the contiguous set of top matches (starting from the 2nd best match) within a distance of $0.5\delta$ are merged. The merge process involves connecting the new node to all the neighbors of the nodes being combined. The counters for each edge are set to the minimum of all counters for the connection to a target node. The number of times a node won is updated as the total number of times the other nodes won. Finally, all these nodes are removed and replaced by a single node that is the weighted average of all the merged nodes. The weights are based on the number of times a particular node has been

**Algorithm 2:** Merge Clusters

**Result**: $G = (E, V)$, AgeMatrix: Staleness of an edge, WinningTimes: # of times $v \in V$ was a winning vector
**Data**: $G = (E, V), AgeMatrix, WinningTimes$

initialize P={}, M={};
**forall the** $v \in V$ **do**
  **forall the** $v' \in V \setminus \{v\}$ **do**
    Let $visual\_distance = \| W_v - W_{v'} \|$;
    Add $(v', visual\_distance)$ to $P$;
  Sort $P$ by $visual\_distance$;
  **forall the** $(v', visual\_distance) \in P$ **do**
    **if** $H(L_v, L'_v) < 0.5 \times \delta$ **then**
      add $v'$ to $M$;
    **else**
      break;
  **if** *$M$ is not empty* **then**
    Let $n$ be the average of all $W_f$ and $L_f \in M$;
    Add $n$ to $E$;
    Change all edges originating from $M$ to originate from $n$;
    Reset $AgeMatrix$ counters for those edges;
    Let $E = E \setminus M$;

a winner. This process is summarized in Algorithm 2 and an example is shown in Figure 2.

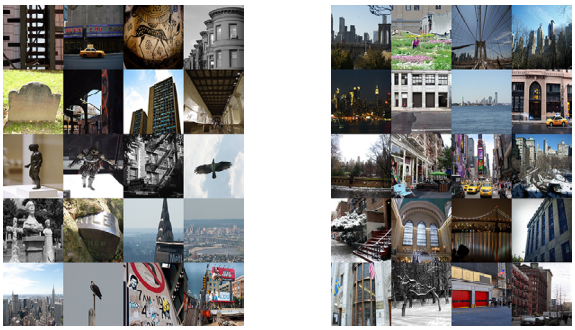
### D. Querying LASOM

**Algorithm 3:** Query LASOM

**Result**: $Location$
**Data**: $G = (E, V)$, $f$: Query Feature

$c_1 \leftarrow \text{argmin}_{c \in E} \| f - W_c \|$;
Let $N$ be all the neighbors of $c_1$;
Let $errors = \{\| f - W_v \| : \forall v \in N \bigcup \{c_1\}\}$;
$errors := (errors - mean(errors))/std(errors)$;
Let $w = 1.5^{-(errors + min(errors))}$;
$w \leftarrow w/\Sigma(w)$;
$Location = 0$;
**forall the** $v \in N \bigcup \{c_1\}$ **do**
  $Location \leftarrow Location + w_v \times L_v$

To assign a location to an untagged image, the best matching node, $c_1$, is chosen first. The codebook vector $c_1$ represents a region in feature space that stops half-way toward any neighboring node, due to the winner-take-all strategy. Therefore the location of the query feature can be anywhere in $c_1$'s area of influence. If this region is large then large errors are possible. To provide better estimates, the query feature is once again compared to $c_1$ and all its immediate neighboring nodes. These errors are then converted to a standard Gaussian distribution by subtracting the mean and dividing by the standard deviation. After shifting the error values such that the minimum value is at 0, an exponential function is used to calculate the weights for each node. The weights are normalized and the location is calculated as a weighted mean of codebook locations. Experiments showed that using an exponential with a base of 1.5 resulted in superior performance. This is summarized in Algorithm 3. An example is shown in Figure 3.



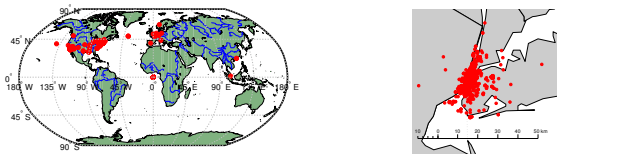
(a) Examples of Training Images (b) Examples of Test Images

Fig. 4: The training set contains much ambiguity. The test set was chosen such that they contain location relevant data.

## IV. Experiments

To evaluate LASOM, we downloaded a set of images from Flickr.com and extracted a number of features. This procedure is described in the next subsection.

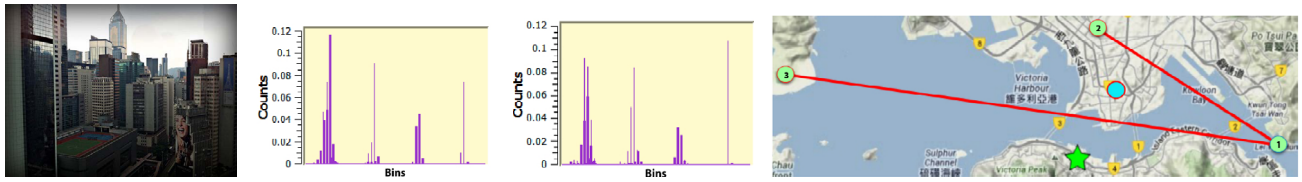
### A. Dataset and features



(a) Distribution of Training Images (b) Distribution of Test Images

Fig. 5: The spatial distribution of images. The test set is almost exclusively taken in New York City, while the training set is noisy and contains a number of distant training samples (Best viewed zoomed in and in color).

Our dataset was constructed by downloading 71,250 images from Flickr.com using location-specific keyword 'NewYorkCity'. The features from all these images have been extracted using high performance computing clusters. 3,011 images from this set were selected at random and filtered for bad images (e.g. black and white or artistic) to create a test set of 1,683. The remaining 2,828 images were discarded. Further 17,508 images were removed from the remaining collection because they were taken by the same photographers whose photos appear in the test set. This resulted in a training set of 49,231 images. A small number of example images from each set are shown in Figure 4. As can be seen, many training images do not contain location relevant information. Although most of the images inside the test and training sets were taken in the New York City area (within 200 km), a number of training images originated in remote locations. Figure 5 shows the location variance.




(a) Test Image (b) Query Color Histogram (c) Best Matching Color Hist. (d) Matching Graph

Fig. 3: Example of LASOM matching a test image (a) based on its color histogram (b) and the color histogram stored in its dictionary (c). (d) Example of how LASOM geotags an images. The test point is the green start at the bottom of the map. The ranking of LASOM codebook vectors in this subgraph is denoted by the number inside the green circles. The estimated location is represented by the cyan circle outlined in red.

Six features were extracted from each image: a 232 bin histogram of edge lengths and edge angles, two vectors of pixel values generated from a 5×5×3 and 16×16×3 pixel version of the image, a 600 dimensional vector representing the global structure, or 'gist' [20], of the image, a 512 bin histogram that captures the distribution of textures, or 'textons', and a 14×14×4 bin CIE L*a*b color histogram. LASOM was trained on each feature independently. The order of training image presentation was randomized for each feature. All histograms were compared using the $\chi^2$ method. $L_1$ distance was used for the other 3 features.

### B. Evaluation

The purpose of applying an algorithm such as LASOM to the geotagging problem is to reduce the number of features that need to be examined for geotagging unannotated images, lower the human labeling burden, decrease sensitivity to noise, and uncover inter-cluster relationships that can be used to improve performance. For these reasons the k-NN approach described in [5] will be used as a baseline. By identifying similar locations, LASOM estimates locations that do not belong to a cluster center. On-line k-means was not considered, because it does not provide such information. The experiments in this section serve to highlight the benefits of LASOM over other SOM approaches, demonstrate that the compression does not effect geolocation accuracy, and that building a graph using an on-line method incurs less computational cost than comparing every pair of images.

### C. Comparing with other SOM algorithms.

In this section we compare LASOM to other popular unsupervised on-line methods, ESOINN and GNG, using the 'New York City' dataset. When learning a distribution of a feature we set the distance threshold, $\delta$, to 2 km except for color which used a $\delta$ of 3 km. The leaning rate, $\epsilon(t)$, was set to $\frac{1}{4t}$ . Cluster merging occurred every $\lambda = 7,500$ trials. For ESOINN, the noise reduction step was performed every 7,500 training samples. ESOINN performed its noise reduction step every 7,500 training samples as well. The density threshold parameters, $c_1$ and $c_2$ are used in ESOINN determine when to delete nodes with two or one neighbors. These parameters were set to 1e-4 and 1e-3, respectively, to discourage node deletions. It was found, however, that these parameters did not effect the behavior of ESOINN significantly, as most nodes were deleted because they had

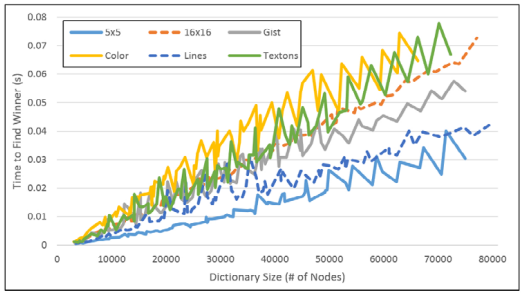
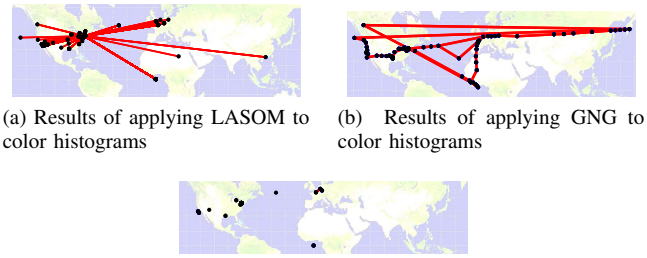



Fig. 6: Time for finding the best matching node in a LASOM map as a function of map size.

no neighbors. For all algorithms, the edge age threshold was set to 10,000 trials. GNG was not allowed to grow past 4,000 codebook vectors and a new vector was added every 50 trials. Training LASOM on each feature required approximately 20 minutes of training time. Figure 6 shows the time required per iteration as the size of the graph increases.



(a) Results of applying LASOM to color histograms (b) Results of applying GNG to color histograms

(c) Results of applying ESOINN to color histograms

Fig. 7: The spatial maps generated by different SOMs. Each node in the graph represents a codebook vector that has visual data associated with it. (a) The codebook vectors for LASOM generated maps that placed most of the codebook vectors in the New York City region. (b) GNG has distributed the codeook vectors to areas unlikely to contain relevant information. (c) ESOINN learns a similar map as LASOM but looses similarity information (fewer edges).

*1) Graph Building:* Figure 7 shows an example of the kind of maps different SOM algorithms learn. LASOM not only learns a compact representation similar to ESOINN, but also stores enough information about visual features, to obtain good geotagging performance. Figure 8 shows some examples of the similarities that are discovered by LASOM when applied to a global image database.

(a) Color Histograms: Bad Lighting

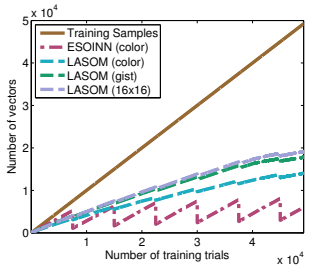
(b) Color Histograms: Trees

(c) Gist

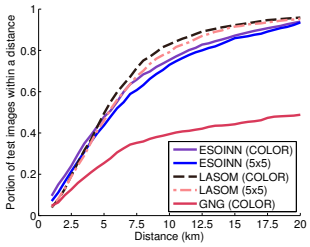
Fig. 8: A neighborhood from LASOM maps and up to 5 images that map to different nodes. Each node represents a set of images and each edge denotes color similarity between different regions. (a) The images taken in poor lighting conditions all form a global similarity network. (b) This subgraph captures structures surrounded by trees. (c) The gist feature describes the structural information of an image, and is better at capturing semantic similarity than color histograms.

According to the figure, GNG appears to be most sensitive to outliers (e.g. the (0,0) point near South Africa). The outliers generate large errors and GNG dedicates many codebook vectors trying to reduce them. ESOINN is most robust in resisting noise, capable of isolating correlated but distant input samples into separate clusters. This can be seen in Figure 7 by comparing how differently GNG and ESOINN react to the images that are incorrectly geotagged as (0,0). ESOINNs robustness is achieved by treating most input as noise. The GPS coordinates are an easier signal to model than the distribution of high dimensional features, therefore spatial coordinates dominate the learning process. In other words, the images might be spatially separate but not separated in visual feature space (e.g. color histogram). If there exists an edge between two distant locations that are not removed during the noise reduction process, then it could either mean that an image was mislabeled or that the two regions share those features. LASOM was more likely to update and merge existing vectors than ESOINN, which did not connect most nodes to any other node.

Over the course of training the local noise reduction rule removed only 581 units from the color histogram LASOM, while the global noise reduction step merged 3,719 codebook




(a) Map growth comparison




(b) Geotagging Accuracy

Fig. 9: Comparing the performance of different SOM algorithms. (a) Comparing codebook vector growth of ESOINN and LASOM algorithms. (b) Geotagging accuracy of test images using color histograms. The discontinuities correspond to the execution of the noise reduction step.

| Alg. | Color | Texture | 5×5 | 16×16 | Gist |
|---|---|---|---|---|---|
| LASOM | 14,032 | 17,613 | 19,144 | 19,144 | 17,803 |
| ESOINN | 6,009 | 6,016 | 4,445 | 4,445 | 4,514 |

TABLE I: The size of the codebook vectors used by LASOM maps. Different features have different spatial sensitivity and therefore require different dictionary sizes.

vectors. The low usage of the local rule is most likely due to the high age threshold setting of 10,000. Furthermore, every time nodes were merged those counters were reset, allowing some nodes generated by noisy data to remain. Figure 9a shows how the number of nodes grows as a function of training signals. The codebook size of LASOM is more than twice as large as ESOINN, because LASOM does not yet have a mechanism for mass removal of nodes. This is the reason LASOM can outperform ESOINN as more information is preserved.

The number of codebook vectors for different maps varied considerably. These values indicate how view-sensitive a feature is. For example, 16×16 images appear to vary greatly in a 2 km region and therefore many codebook vectors are required to represent such a large area, while a single codebook vector for texture map is able to describe more area. Color histograms, which were trained using a larger distance threshold, 3 km, required an even smaller dictionary. Table I summarizes these results.

*2) Geotagging Accuracy:* Figure 9b shows that LASOM learns a better representation of how visual features are related to one another. ESOINN performed well on this dataset and stored more codebook vectors than expected, suggesting that ESOINN could learn a good representation of the world if it is trained on small regions for long periods. However, LASOM trained on 5×5 images begins to outperform ESOINN at a distance of a little over 4 km and the color histograms LASOM gets better at about 5 km. This difference is consistent with our expectations of how the distance threshold should effect performance. The 2 km threshold distance used by the 5×5 LASOM should on average introduce errors of about 2 km and can be as far as 2 km from the training sample. The 3 km threshold used in color LASOM increase the possible error to 6 km.

### D. Discussion

Solving the geolocation problem requires compact data structures that allow for efficient image retrieval. The more relevant the retrieved image is to a query image the more likely the GPS coordinates assigned to it are close to the query image as well. One of the insights we make about this problem is that the actual images do not need to be stored, and a compact representation suffices as long as all the important locations signatures are preserved. Self-Organizing Maps is an efficient way to compress a large database to a few discrete units.

Methods like GNG and ESOINN do not handle sparse and highly variable data very well and require manual tuning the weight between the feature vector and the GPS coordinates to prevent one from dominating the other. Although ESOINN performed quite well on this dataset, when applied to a dataset of over one million images, it produced a small map of a little over 700 codebook vectors spread over 52 distinct clusters. It is also interesting to see that different features produce about the same codebook sizes (Table 1) meaning that ESOINN does not explain the visual feature variability very well, while LASOM allows us to evaluate the usefulness of different features. This reveals that ESOINN is extremely sensitive to how it is trained. We expect bigger differences between ESOINN and LASOM once larger datasets are used.

LASOM also performed quite well, while at the same time achieving high compression rates, as large as 3.6. The spatial constraints used in the experimental section define histogram bin widths in terms of spatial distance. Within those bins LASOM learns the distribution of features and their likelihood. If a number of separate distributions are similar than they are merged, otherwise, a multi-modal distribution is stored. By examining the topology, it can be determined how far this feature lies from the cluster center and interpolate its coordinates. From the results it is evident that LASOM can to handle the sparseness of the training set.

Currently, LASOM assumes that the visual features are the only source of noise. However, as we saw in Figure 5 there can be any number of erroneously geotagged images in our training set (the point at (0,0) near South Africa). One way to address this problem is by comparing the number of times a codebook vector was the best node with the number of times it was the second best. If this difference is large, but the best winning node to which it is connected is far away, then that could serve as evidence that the image contains an incorrect geotag.

## V. Conclusion

There are a number of limitations in brute-force methods for solving the geolocation problem, including sensitivity to noise and high computational and storage costs. The topological relationship between visual features and locations provides an important source of context for the geotagging problem. We have shown that such graphs can be constructed using an on-line unsupervised Location Aware Self Organizing Map (LASOM). Furthermore, we demonstrated on a real-world dataset collected from Flickr.com that this approach can be used for solving the geolocation problem. The location awareness provides new research opportunities. For example, the distance between neighboring codebook vectors can be used to build and evaluate multiple location hypotheses. We plan to further examine this relationship between different distance thresholds, compression performance and localization accuracy.

## References


[1] L. Cao, J. Luo, A. Gallagher, X. Jin, J. Han, and T. Huang, "A worldwide tourism recommendation system based on geotaggedweb photos," in *ICASSP*, 2010, pp. 2274–2277.

[2] Y.-H. Kuo, W.-Y. Lee, W. H. Hsu, and W.-H. Cheng, "Augmenting mobile city-view image retrieval with context-rich user-contributed photos," in *Proc. of the 19th ACM Multimedia*. ACM, 2011, pp. 687–690.

[3] Y. Zhou and J. Luo, "Geo-location inference on news articles via multimodal plsa," in *Proc. of the 20th ACM Multimedia*. ACM, 2012, pp. 741–744.

[4] Y.-C. Song, Y.-D. Zhang, J. Cao, T. Xia, W. Liu, and J.-T. Li, "Web video geolocation by geotagged social resources," *Multimedia, IEEE Transactions on*, vol. 14, no. 2, pp. 456–470, 2012.

[5] J. Hays and A. Efros, "IM2GPS: estimating geographic information from a single image," *CVPR*, 2008.

[6] S. Cao and N. Snavely, "Graph-based discriminative learning for location recognition," *CVPR*, 2013.

[7] D. Kit, B. T. Sullivan, and D. H. Ballard, "Novelty detection using growing neural gas for visuo-spatial memory," in *IROS*. IEEE, 2011, pp. 1194–1200.

[8] O. A. B. Penatti, L. T. Li, J. Almeida, and R. da S. Torres, "A visual approach for video geocoding using bag-of-scenes," in *Proceedings of the 2Nd ACM Int'l. Conf. on Multimedia Retrieval*, ser. ICMR '12. New York, NY, USA: ACM, 2012, pp. 53:1–53:8.

[9] M. Cristani, A. Perina, U. Castellani, and V. Murino, "Geo-located image categorization and location recognition," *Pattern Recognition and Image Analysis*, vol. 19, no. 2, pp. 245–252, 2009.

[10] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Springer, 2001.

[11] D. J. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg, "Mapping the world's photos," in *Proc. of the 18th Int'l Conf. on World Wide Web*. ACM, 2009, pp. 761–770.

[12] E. Kalogerakis, O. Vesselova, J. Hays, A. Efros, and A. Hertzmann, "Image sequence geolocation with human travel priors," in *ICCV*, 2009, pp. 253–260.

[13] D. Joshi, A. Gallagher, J. Yu, and J. Luo, "Exploring user image tags for geo-location inference," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE Int'l Conf. on*, 2010, pp. 5598–5601.

[14] C. Sengstock and M. Gertz, "Latent geographic feature extraction from social media," in *Proc. of the Int'l Conf. on Advances in Geographic Information Systems*, ser. SIGSPATIAL. ACM, 2012, pp. 149–158.

[15] P. Serdyukov, V. Murdock, and R. van Zwol, "Placing flickr photos on a map," in *Proceedings of the 32Nd Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, ser. SIGIR '09. New York, NY, USA: ACM, 2009, pp. 484–491.

[16] J. Luo, D. Joshi, J. Yu, and A. Gallagher, "Geotagging in multimedia and computer vision - a survey," *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 187–211, 2011.

[17] B. Fritzke, "A growing neural gas network learns topologies," in *NIPS 7*. MIT Press, 1995, pp. 625–632.

[18] S. Furao, T. Ogura, and O. Hasegawa, "An enhanced self-organizing incremental neural network for online unsupervised learning," *Neural Networks*, vol. 20, no. 8, pp. 893 – 903, 2007.

[19] F. Shen and O. Hasegawa, "Self-organizing incremental neural network and its application," in *ICANN 2010*, ser. Lecture Notes in Computer Science, K. Diamantaras, W. Duch, and L. Iliadis, Eds. Springer Berlin Heidelberg, 2010, vol. 6354, pp. 535–540.

[20] A. Oliva and A. Torralba, "Chapter 2 building the gist of a scene: the role of global image features in recognition," in *Visual Perception Fundamentals of Awareness: Multi-Sensory Integration and High-Order Perception*, ser. Progress in Brain Research. Elsevier, 2006, vol. 155, Part B, pp. 23 – 36.