

LiveMaps - Converting Map Images into Interactive Maps

Michael R. Evans
Microsoft Corporation
Sunnyvale, CA 94086
mievans@microsoft.com

Dragomir Yankov
Microsoft Corporation
Sunnyvale, CA 94086
dragoy@microsoft.com

Pavel Berkhin
Microsoft Corporation
Sunnyvale, CA 94086
pavelbe@microsoft.com

Pavel Yudin
Microsoft Corporation
Moscow, Russia
payudin@microsoft.com

Florin Teodorescu
Microsoft Corporation
Bellevue, WA 98004
florint@microsoft.com

Wei Wu
Microsoft Corporation
Bellevue, WA 98004
weiwu@microsoft.com

ABSTRACT

Image search is a popular application on web search engines. Issuing a location-related query in image search engines often returns multiple images of maps among the top ranked results. Traditionally, clicking on such images either opens the image in a new browser tab or takes users to a web page containing the image. However, finding the area of intent on an interactive web map is a manual process. In this paper, we describe a novel system, *LiveMaps*, for analyzing and retrieving an appropriate map viewport for a given image of a map. This allows annotation of images of maps returned by image search engines, allowing users to directly open a link to an interactive map centered on the location of interest.

LiveMaps works in several stages. It first checks whether the input image represents a map. If yes, then the system attempts to identify what geographical area this map image represents. In the process, we use textual as well as visual information extracted from the image. Finally, we construct an interactive map object capturing the geographical area inferred for the image. Evaluation results on a dataset of high ranked location images indicate our system constructs very precise map representations also achieving good levels of coverage.

CCS CONCEPTS

•Information systems → Image search; Information extraction;

KEYWORDS

Image Search; Map Search; Geographic Information Retrieval

ACM Reference format:

Michael R. Evans, Dragomir Yankov, Pavel Berkhin, Pavel Yudin, Florin Teodorescu, and Wei Wu. 2017. LiveMaps - Converting Map Images into Interactive Maps. In *Proceedings of SIGIR'17, August 7-11, 2017, Shinjuku, Tokyo, Japan.*, 4 pages.
DOI: <http://dx.doi.org/10.1145/3077136.3080673>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGIR'17, August 7-11, 2017, Shinjuku, Tokyo, Japan

© 2017 ACM. 978-1-4503-5022-8/17/08...\$15.00

DOI: <http://dx.doi.org/10.1145/3077136.3080673>

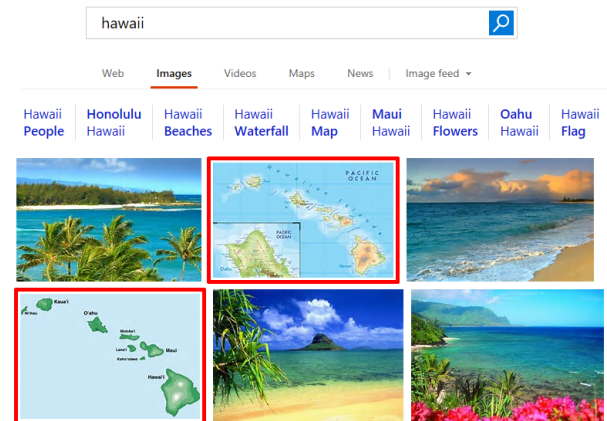


Figure 1: Images of maps on web image search.

1 INTRODUCTION

In this paper, we describe *LiveMaps* – a system which converts images of maps into interactive maps by opening an online map engine to the area of intent located in the original map image. We are developing this system for use on web image search as many queries which users issue have location intent (e.g., names of countries, states, parks, tourist destinations). Frequently, top results returned by image search contain images of maps, as shown in Figure 1. It is reasonable to assume that if a user clicks on a map image, their intent is to explore the map depicted on it in more detail, which may include zooming in to see more detail, or scrolling to see neighboring context. If users want to interact with the maps returned by major image search engines, they are forced to locate the area of interest on a separate interactive map engine. A user may attempt to simply issue the same query on the map endpoint as they did on the image endpoint, and sometimes they will get a relevant map viewport, which we use as a baseline in our experiment section. However, in many situations, the user query does not contain enough context to recreate the area of geo-intent contained in the map image. We propose a specialized system that will annotate images of maps with map viewport information, allowing a user to directly open an interactive map with the relevant area of geo-intent.

In addition to image search, this system has use cases across the web. Many news stories and articles discuss locations - news about international events, economic news, weather forecasts, etc. It is

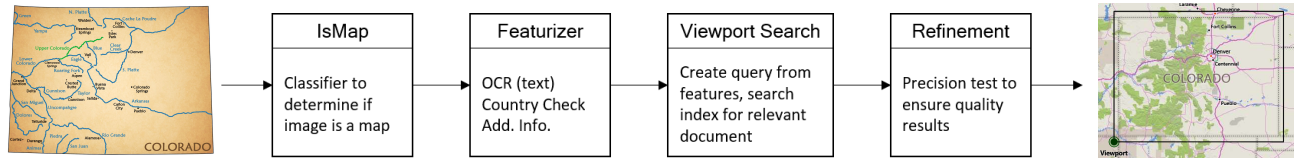


Figure 2: Input/Output and LiveMaps architecture.

common to see map images embedded in such articles. Readers may wish to acquire additional context by exploring closely the location depicted on the map - zooming in or out, moving the map viewport around. This cannot be achieved with the original image and an interactive map is needed.

Determining the area of geo-intent (a *viewport*) from a static map image is challenging due to the inherent noise in analyzing the image and the sheer size of the search space, ranging from continents down to postal codes. In addition, the query used to find the image of the map may not contain enough information to properly locate the area of intent. We propose novel solutions to address these challenges.

Related work in the area focuses on geo-location inference from pictures and related metadata [3]. Frequently geo-location determination is focused on the metadata around an image (e.g., geo-tags, textual content associated with an image, and referring URLs) [6]. Our work is more related to map digitization [7], in that we attempt to understand the map image itself, rather than visually recognizing landmarks, or skylines [5].

The paper format is as follows. In Section 2, we describe the components of the *LiveMaps* system. In Section 3, we experimentally validate our system using hand-labeled images from human judges against a robust baseline. Finally, in Section 4, we summarize our system and describe future work.

2 SYSTEM ARCHITECTURE

LiveMaps is comprised of several components, as shown in Figure 2. Given an image, *LiveMaps* first classifies whether the image appears to be of a map. This is done via an image-based classifier we call the *IsMap* classifier, explained in Section 2.1. If the image is not a map then we simply exit without producing any map annotation for the image. If the image is detected to be of a map, *LiveMaps* begins to *Featurize* the image as explained in Section 2.2. This consists of a set of sub-components that derive features from the image. The output is a set of features which we combine into a ‘meta query’ that we pass to our *Viewport Search Engine* (detailed in Section 2.3) to identify the *area of geo-intent* depicted by the image. The index contains text documents representing all administrative boundaries in the world (e.g., countries/cities/states). The documents also contain the spatial shape of the location they represent. This shape is used to construct the map viewport ultimately returned by the system. Finally, to ensure high precision, *LiveMaps* also runs a *Refinement* component which looks at the original image and the derived map and decides purely visually whether the two look similar. This is achieved with a Image-Map Similarity detection classifier. If the classifier says with high confidence that the image and the derived map are similar we proceed and annotate the image with our map. If not we exit without producing any annotation.

2.1 IsMap Classifier

The *IsMap* classifier is a binary classifier which uses image analysis to detect whether an image represents a map. We trained a deep neural network model with *CNTK*¹ using a pre-trained residual network of eighteen layers (*ResNet-18* [4]). We modified the last layer to have a softmax binary output - ‘map’ or ‘not map’. As training data we used approximately 1,500 images of maps annotated from the publicly available *ImageNet* [1] dataset. We also sample uniformly at random another fifty thousand non-maps using the same set. All inputs are resized to 256x256 using three (RGB) channels for depth. Figure 3 shows a precision-recall curve of the model computed on a judged annotate test set \mathcal{D}_T which we describe in more details in our evaluation section.

2.2 Featurizer

Given an image, we extract several types of features. Not all features may be present for every map image. The primary feature comes from the textual information derived from optical character recognition (OCR) of the image. Terms from OCR and other extractors are combined into a query. The terms in the query can be weighted based on our confidence in the accuracy of each extractor. If there are additional extractors, e.g. image name or text surrounding the image, we can simply concatenate their output to the query already constructed by the current extractors. At present we have two extractors - OCR and Country Detection.

OCR Feature Extractor. Whenever a map image contains machine readable text, that text becomes a very strong predictor for the area of geo-intent depicted by the image. We extract the text from the image with a publicly available OCR library². For many images, however, there are numerous problems: the text may be blurred; OCR may break or merge some terms incorrectly; there may be many irrelevant image terms, such as map legends. All these considerations make the OCR input potentially noisy. We apply a few filtering rules to eliminate the noisiest of text, however, frequently the output will contain potentially a majority of OCR’ed terms that are not relevant to the image. Regardless, this text is used as the query for matching potential viewports in our *Viewport Search* engine, discussed below. Improving OCR for non-traditional use cases is outside the scope of this paper.

Country Extractor. Many countries have distinctive border shapes or coastal lines which facilitates learning to detect them visually. We built a classifier to predict the presence of a country as follows.

We constructed a training set \mathcal{D}_C with automatically labeled images for each of 250 countries by scraping queries of the form {map of *country_i*}. We issue these queries against an image search

¹<https://github.com/microsoft/cntk>

²<https://www.microsoft.com/cognitive-services/>

engine and collect the top 150 results per query. We split the dataset into a training part, approximately 120 images per country, and a testing part, the remaining images. Note, we make sure that none of the images in the training set are also in the measurement set we use later in our experimental evaluation.

Using \mathcal{D}_C , we trained a country detection classifier. The classifier again uses purely visual inputs. Similar to the *IsMap* classifier, we use a pre-trained residual network model of eighteen layers. The input is again images of size and depth 256x256x3. The output layer in this case is a softmax over 250 categories - one for each of the top 250 countries and territories in the world. If the model detects as top category country X with score larger than 0.80, then we add the top country to the overall meta query which we construct. This feature extractor is especially important when there is no text or the text is too little or blurred for the OCR extractor to read it correctly (Fig 3). At present the model is trained only to detect countries. We are currently augmenting it with other locations with distinctive shapes, e.g. islands or states.

2.3 Viewport Search Engine

Once an image is featurized, we generate a query for our Viewport Search Engine. We utilize standard search engine technology (referred to as our ‘index’) for retrieving and ranking documents. Our assumption is that the featurizers from above will extract textual information from the map image indicative about the administrative entity it depicts, and that we can retrieve a document about that same administrative entity with said textual information. For example, if an image contains the terms “San Francisco, Oakland, Los Angeles, Sacramento, San Diego”, presumably the highest ranking document returned from our index will be the document for California. The documents contained in our index represent each administrative district worldwide (e.g., a document for the United States, one for California, one for San Francisco). Each document contains textual information representing the contents of that administrative boundary (e.g., the United States document has a list of popular cities and all states). The document also contains the geometry of the entity, as well as a bounding box, which is passed to the Refinement step.

In the matching phase, we generate a query from the OCR text. The index finds documents with some minimum number of matching query terms, and in the ranking phase, the text from an image will be matched and featurized for ranking the final list of documents. The index uses a machine-learned model we trained (Gradient Boosted Regression Trees [2]) for ranking of documents on a number of features: administrative level (country/city/state/etc), number of matching phrases, TF-IDF scores, etc. For training we use the 2K locations discussed in the next section with their viewports as returned by the map search engine. We then apply a threshold to determine whether the top ranking document should be passed to the next step for further refinement or if we should exit and return nothing.

2.4 Refinement

If the system generates a map which does not correctly represent the area of geo-intent from the image, this leads to a poor user experience. To avoid this and improve the precision of the system,

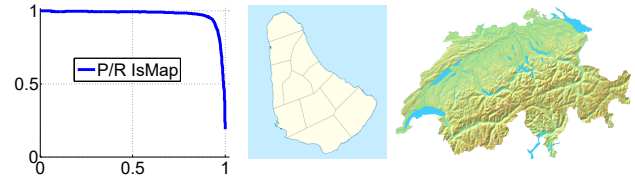


Figure 3: Left: P/R curve for the *IsMap* model. Center: Country classifier, top category and score: (Barbados:0.982); right: (Switzerland:0.963).

we built a refinement component. This component runs an *Image-Map Similarity* classifier: the classifier looks at the original image and the map that was generated on the previous step and compares them visually. If the two are found to be similar, we proceed with the process, otherwise *LiveMaps* exits without returning a map to the user. To compare a map to an image, we first convert the map itself into an image using a web-based map API. Now the problem reduces to one of comparing whether two images are similar.

Training data: to train the *Image-Map Similarity* model, we construct a dataset of pairs of images. For each of 2K location entities (e.g. countries, cities, states, islands) collected from Wikipedia, we construct a query $q_i = \{\text{map of } location_i\}$. We issue the queries against a web image search scraping the top 5 results, e.g. for query q_i we obtain s_i^j ($j = 1..5$). We then issue the queries against a web map search API³ obtaining one map image m_i . We now construct pairs. A pair has a positive label (m_i, s_i^j) = 1 if both were scraped by the same query q_i , e.g. both are likely to represent {France}. This gives us approximately 10K positive pairs. We also generate negative pairs by matching (m_i, s_k^j) = 0 - the map search result for a query q_i with the image search result for a query q_k . We sample 50K such negative pairs.

Training method: From each pair we construct input of size and depth 256x256x6, i.e. using six depth channels, three from each image in the pair. We train a convolutional neural network (CNN) with two convolutional layers followed by MaxPooling and a dense layer with Dropout. The output layer is a softmax binary output predicting whether the map and the image are ‘similar’ or ‘not-similar’. This step significantly increases precision at the cost of some recall.

3 EVALUATION

Datasets. To evaluate the end-to-end precision and recall of our system we use the following datasets.

³<https://msdn.microsoft.com/en-us/library/ff701713.aspx>

	Baseline	LiveMaps			
		$\theta \geq 0.0$	$\theta \geq 0.1$	$\theta \geq 0.5$	$\theta \geq 0.9$
Precision	0.58	0.86	0.89	0.91	0.93
Recall	0.54	0.17	0.14	0.09	0.05
Avg S	0.30	0.45	0.46	0.47	0.50

Table 1: Results on the \mathcal{D}_T^M dataset where $S_i \geq 0.2$ and θ is a threshold for the *Image-Map Similarity* model scores.

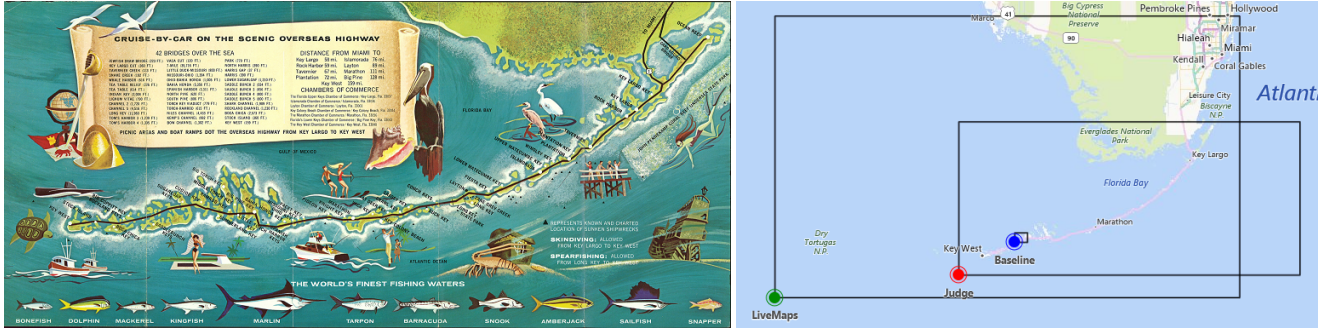


Figure 4: Left: Source map image. Right: Viewports of Baseline, Human Judge, and LiveMaps. We measure correctness via Equation 1 against the judge’s viewport. LiveMaps: $s_i = 0.307$, Baseline: $s_i = 0.002$.

$\mathcal{D}_{\mathcal{T}}$: Constructed by sampling 20K queries from the image search traffic over a period of several months. Judges labeled 670 of the queries as having ‘location intent’, among them queries such as {Acadia national park}, {island of Oahu}, etc. For each of the 670, queries we scraped the top 30 results from image search. We thus obtained a dataset of approximately 20K images. The set was labeled by judges who identified 3,500 images from it as maps images.

$\mathcal{D}_{\mathcal{T}}^M$: The subset of 3,500 maps images from the $\mathcal{D}_{\mathcal{T}}$ test set. We asked the judges to use map search and capture the corner coordinates of the map viewport best representing the map from the image. The evaluation results in this section are computed on this set using the viewports derived from the judges.

Metric. For an image $i \in \mathcal{D}_{\mathcal{T}}^M$, let V_i^J be the map viewport created by our human judges and V_i^L be the viewport as provided by *LiveMap*. We compute a measure of quality of the system by comparing the extent of overlap between its viewports and the judge’s viewport. In particular, we compute the quantity:

$$S_i = \frac{V_i^J \cap V_i^L}{V_i^J \cup V_i^L} \quad (1)$$

When $V_i^J \approx V_i^L$ we have $S_i \rightarrow 1$, and when $V_i^J \cap V_i^L = \emptyset$ we have $S_i = 0$.

Translations in the viewport, or in zoom level can cause drop in the similarity score S_i , but the derived viewports may still be acceptable. Based on multiple observations we see that an overlap of $S_i \geq 0.2$ indicates a viewport which judges would consider an acceptable representation of the original map image. Figure 4 shows an example where the judged viewport (rectangle with the word ‘Judge’ in lower left-hand corner) has similarity of $S_i = 0.307$ with the *LiveMaps* viewport (labeled ‘LiveMaps’). Both viewports capture the area of geo-intent from the image, the Florida keys, but judges have used a tighter zoom level than the system. While the Judge’s viewport is more accurate to the original image, *LiveMaps* still returns an acceptable result. *LiveMaps* could be improved by cropping the returned administrative bounding box (viewport) to more accurately represent the image.

Baseline. In the core of the $\mathcal{D}_{\mathcal{T}}^M$ dataset are image search queries. A natural question is: If the image search query is, say {France}, should we forward it to maps search and display whatever map is returned? We refer to this as the *Baseline*. As we will

show, this works some of the time, but the precision is unusably low.

Results. As stated, we assume that if for an image i we have $S_i \geq 0.2$ then we have constructed a viewport which accurately represents the image, i.e., we count this example as a true positive in computing precision and recall. Table 1 shows the performance of *LiveMaps* on the $\mathcal{D}_{\mathcal{T}}^M$ dataset and compares it with *Baseline*. For *LiveMaps*, we vary the threshold θ over the scores of the *Image-Map Similarity* model, where a threshold of 0 indicates the similarity model isn’t being used - in which case the average S is 0.45. For instance, if we set $\theta \geq 0.5$ we return a viewport for an image only if the *Image-Map* similarity model says that they are similar with score more than 0.5. For the images that pass the similarity threshold and a minimum index document score we compute the metric S_i , and for the rest we say that we have not recalled them. We assume that all queries have valid answers in our index, and therefore recall is computed over all queries. As can be seen from the table, if we impose very strict requirements on the visual similarity between image and derived map ($\theta \geq 0.9$) the system achieves very high precision (number of returned viewports with metric $S_i \geq 0.2$).

4 CONCLUSION

In this paper, we propose a novel system for annotating images of maps with their areas of geo-intent, to be used for opening interactive map engines on the area of interest matching the original image. The system achieves high precision, as is required for a user-facing service, and our next steps will be research into more feature extractors to improve recall.

REFERENCES

- [1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [2] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 2001.
- [3] A. Gallagher, D. Joshi, J. Yu, and J. Luo. Geo-location inference from image content and user tags. In *CVPR Workshops 2009*, pages 55–62. IEEE, 2009.
- [4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR16*, 2016.
- [5] H. Liu, T. Mei, J. Luo, H. Li, and S. Li. Finding perfect rendezvous on the go: accurate mobile visual localization and its applications to routing. In *ACM Multimedia '12*, pages 9–18. ACM, 2012.
- [6] J. Luo, D. Joshi, J. Yu, and A. Gallagher. Geotagging in multimedia and computer vision - survey. *Multimedia Tools and Applications*, 51(1):187–211, 2011.
- [7] H. Samet and A. Soffer. Marco: Map retrieval by content. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):783–798, 1996.