Data Analysis Project: What Makes Paris Look like Paris?

Carl Doersch*
Machine Learning Department
Carnegie-Mellon University
Pittsburgh, PA 15213
cdoersch@cs.cmu.edu

Abstract

Given a large repository of geotagged imagery, we seek to automatically find visual elements, e.g. windows, balconies, and street signs, that are most distinctive for a certain geo-spatial area, for example the city of Paris. This is a tremendously difficult task as the visual features distinguishing architectural elements of different places can be very subtle. In addition, we face a hard search problem: given all possible patches in all images, which of them are both frequently occurring *and* geographically informative? To address these issues, we propose to use a discriminative clustering approach able to take into account the weak geographic supervision. We show that geographically representative image elements can be discovered automatically from Google Street View imagery in a discriminative manner. We demonstrate that these elements are visually interpretable and perceptually geo-informative. The discovered visual elements can also support a variety of *computational geography* tasks, such as mapping architectural correspondences and influences within and across cities, finding representative elements at different geo-spatial scales, and geographically-informed image retrieval.

1 Introduction

Consider the two photographs in Figure 1, both downloaded from Google Street View. One comes from Paris, the other one from London. Can you tell which is which? Surprisingly, even for these nondescript street scenes, people who have been to Europe tend to do quite well on this task. In an informal survey, we presented 11 subjects with 100 random Street View images of which 50% were from Paris, and the rest from eleven other cities. We instructed the subjects (who have all been to Paris) to try and ignore any text in the photos, and collected their binary forced-choice responses (Paris / Not Paris). On average, subjects were correct 79% of the time ($std = 6.3$), with chance at 50% (when allowed to scrutinize the text, performance for some subjects went up as high as 90%). What this suggests is that people are remarkably sensitive to the geographically-informative features within the visual environment. But what are those features? In informal debriefings, our subjects suggested that for most images, a few localized, distinctive elements "immediately gave it away". E.g. for Paris, things like windows with railings, the particular style of balconies, the distinctive doorways, the traditional blue/green/white street signs, etc. were particularly helpful. Finding those features can be difficult though, since every image can contain more than $25,000$ candidate patches, and only a tiny fraction will be truly distinctive.

In this work, we want to find such local geo-informative features *automatically*, directly from a large database of photographs from a particular place, such as a city. Specifically, given tens of thousands of geo-localized images of some geographic region R, we aim to find a few hundred visual elements

*In collaboration with Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei Efros

Figure 1: These two photos might seem nondescript, but each contains hints about which city it might belong to. Given a large image database of a given city, our algorithm is able to automatically discover the geographically-informative elements (patch clusters to the right of each photo) that help in capturing its "look and feel". On the left, the emblematic street sign, a balustrade window, and the balcony support are all very indicative of Paris, while on the right, the neoclassical columned entryway sporting a balcony, a Victorian window, and, of course, the cast iron railing are very much features of London.

that are both: 1) repeating, i.e. they occur often in R, and 2) geographically discriminative, i.e. they occur much more often in R than in R^C. Figure 1 shows sample output of our algorithm: for each photograph we show three of the most geo-informative visual elements that were automatically discovered. For the Paris scene (left), the street sign, the window with railings, and the balcony support are all flagged as informative.

But why is this topic important for modern computer graphics? 1) Scientifically, the goal of understanding which visual elements are fundamental to our perception of a complex visual concept, such as a place, is an interesting and useful one. Our paper shares this motivation with a number of other recent works that don't actually synthesize new visual imagery, but rather propose ways of finding and visualizing existing image data in better ways, be it selecting candid portraits from a video stream [8], summarizing a scene from photo collections [32], finding iconic images of an object [1], etc. 2) More practically, one possible future application of the ideas presented here might be to help CG modelers by generating so-called "reference art" for a city. For instance, when modeling Paris for PIXAR's *Ratatouille*, the co-director Jan Pinkava faced exactly this problem: "The basic question for us was: 'what would Paris look like as a model of Paris?', that is, what are the main things that give the city its unique look?" [26]. Their solution was to "run around Paris for a week like mad tourists, just looking at things, talking about them, and taking lots of pictures" not just of the Eiffel Tower but of the many stylistic Paris details, such as signs, doors etc. [26](see photos on pp.120–121). But if going "on location" is not feasible, our approach could serve as basis for a detail-centric reference art retriever, which would let artists focus their attention on the most statistically significant stylistic elements of the city. 3) And finally, more philosophically, our ultimate goal is to provide a *stylistic narrative* for a visual experience of a place. Such narrative, once established, can be related to others in a kind of geo-cultural visual reference graph, highlighting similarities and differences between regions. E.g. one could imagine finding a visual appearance "trail" from Greece, through Italy and Spain and into Latin America. In this work, we only take the first steps in this direction – connecting visual appearance across cities, finding similarities within a continent, and differences between neighborhoods. But we hope that our work might act as a catalyst for research in this new area, which might be called *computational geo-cultural modeling*.

2 Prior Work

In the field of architectural history, descriptions of urban and regional architectural styles and their elements are well established, e.g. [22, 35]. Such local elements and rules for combining them have been used in computer systems for procedural modeling of architecture to generate 3D models of entire cities in an astonishing level of detail, e.g. [24], or to parse images of facades, e.g. [36]. However, such systems require significant manual effort from an expert to specify the appropriate elements and rules for each architectural style.

At the other end of the spectrum, data-driven approaches have been leveraging the huge datasets of geotagged images that have recently become available online. For example, Crandall et al. [6] use the GPS locations of 35 thousand consumer photos from Flickr to plot photographer-defined frequency maps of cities and countries, while Kalogerakis et al. [15] use the locations and rela-

tive time-stamps of photos of the same photographer to model world-wide human travel priors. Geo-tagged datasets have also been used for place recognition [29, 17, 3] including famous landmarks [20, 21, 38]. Our work is particularly related to [29, 17], where geotags are also used as a *supervisory signal* to find sets of image features discriminative for a particular place. While these approaches can work very well, their image features typically cannot generalize beyond matching specific buildings imaged from different viewpoints. Alternatively, global image representations from scene recognition, such as GIST descriptor [25] have been used for geo-localization of generic scenes on the global Earth scale [14, 15]. There, too, reasonable recognition performance has been achieved, but the use of global descriptors makes it hard for a human to interpret *why* a given image gets assigned to a certain location.

Finally, our paper is related to a line of work on unsupervised object discovery [28, 5, 16, 18, 33] (and especially [27], who also deal with mining geo-tagged image data). Such methods attempt to explicitly discover features or objects which occur frequently in many images and are also useful as human-interpretable elements of visual representation. But being unsupervised, these methods are limited to only discovering things that are both very common and highly visually consistent. However, adding supervision, such as by explicitly training object [19] or object part detectors [2], requires a labor-intensive labeling process for each object/part.

In contrast, here we propose a discovery method that is weakly constrained by location labels derived from GPS tags, and which is able to mine representative visual elements automatically from a large online image dataset. Not only are the resulting visual elements geographically discriminative (i.e. they occur only in a given locale), but they also typically look meaningful to humans, making them suitable for a variety of geo-data visualization applications. The next section describes the data used in this work, followed by the full description of our algorithm.

3 The Data

Flickr has emerged as the data-source of choice for most recently developed data-driven applications in computer vision and graphics, including visual geo-location [14, 6, 21]. However, the difficulty with Flickr and other consumer photo-sharing websites for geographical tasks is that there is a strong data bias towards famous landmarks. To correct for this bias and provide a more uniform sampling of the geographical space, we turn to GOOGLE STREET VIEW – a huge database of street-level imagery, captured as panoramas using specially-designed vehicles. This enables extraction of roughly fronto-parallel views of building facades and, to some extent, avoids dealing with large variations of camera viewpoint.

Given a geographical area on a map, we automatically scrape a dense sampling of panoramas of that area from Google Street View [12]. From each panorama, we extract two perspective images (936x537 pixels), one on each side of the capturing vehicle, so that the image plane is roughly parallel to the vehicle's direction of motion. This results in approximately $10,000$ images per city. For this project, we downloaded 12 cities: Paris, London, Prague, Barcelona, Milan, New York, Boston, Philadelphia, San Francisco, San Paulo, Mexico City, and Tokyo. We have also scraped suburbs of Paris for one experiment.

4 Discovering geo-informative elements

Our goal is to discover visual elements which are characteristic of a given geographical locale (e.g. the city of Paris). That is, we seek patterns that are both *frequently occurring* within the given locale, **and** *geographically discriminative*, i.e. they appear in that locale and do not appear elsewhere. Note that neither of these two requirements by itself is enough: sidewalks and cars occur frequently in Paris but are hardly discriminative, whereas the Eiffel Tower is very discriminative, but too rare to be useful ($< 0.0001\%$ in our data). In this work, we will represent visual elements by square image patches at various resolutions, and mine them from our large image database. The database will be divided into two parts: (i) the positive set containing images from the location whose visual elements we wish to discover (e.g. Paris); and (ii) the negative set containing images from the rest of the world (in our case, the other 11 cities in the dataset). We assume that many frequently occurring but uninteresting visual patterns (trees, cars, sky, etc.) will occur in both the positive and negative sets, and should be filtered out. Our biggest challenge is that the overwhelming majority of our data

(a) K-Means Clusters using SIFT (Visual Words) (b) K-Means Clusters using HOG

Figure 2: (a) k-means clustering using SIFT (visual words) is dominated by low level features. (b) k-means clustering over higher dimensional HOG features produces visually incoherent clusters.

is uninteresting, so matching the occurrences of the rare interesting elements is like finding a few needles in a haystack.

One possible way to attack this problem would be to first discover repeated elements and then simply pick the ones which are the most geographically discriminative. A standard technique for finding repeated patterns in data is clustering. For example, in computer vision, "visual word" approaches [34] use k-means clustering on image patches represented by SIFT descriptors. Unfortunately, standard visual words tend to be dominated by low-level features, like edges and corners (Figure 2a), not the larger visual structures we are hoping to find. While we can try clustering using larger image patches (with a higher-dimensional feature descriptor, such as HOG [7]), k-means behaves poorly in very high dimensions because the distance metric becomes less meaningful, producing visually inhomogeneous clusters (Figure 2b). We also experimented with other clustering approaches, such as Locality-Sensitive Hashing [11], with similar results.

An alternative approach is to use the geographic information as part of the clustering, extracting elements that are both repeated and discriminative at the same time. We have experimented with such discriminative clustering methods [23, 10, 30], but found they did not provide the right behavior for our data: they either produce inhomogeneous clusters or focus too much on the most common visual features. We believe this is because such approaches include at least one step that partitions the *entire* feature space. This tends to lose the needles in our haystack: the rare discriminative elements get mixed with, and overwhelmed by, less interesting patches, making it unlikely that a distinctive element could ever emerge as its own cluster.

In this paper, we propose an approach that avoids partitioning the entire feature space into clusters. Instead, we start with a large number of randomly sampled candidate patches, and then give each candidate a chance to see if it can converge to a cluster that is both frequent and discriminative. We first compute the nearest neighbors of each candidate, and reject candidates with too many neighbors in the negative set. Then we gradually build clusters by applying iterative discriminative learning to each surviving candidate. The following section presents the details of this algorithm.

4.1 Our Approach

From the tens of millions of patches in our full positive set, we randomly sample a subset of $25,000$ high-contrast patches to serve as candidates for seeding the clusters. Throughout the algorithm, we represent such patches using a HOG+color descriptor [7]. First, the initial geo-informativeness of each patch is estimated by finding the top 20 nearest neighbor (NN) patches in the full dataset (both positive and negative), measured by normalized correlation. Patches portraying non-discriminative elements tend to match similar elements in both positive and negative set, while patches portraying a non-repeating element will have more-or-less random matches, also in both sets. Thus, we keep the candidate patches that have the highest proportion of their nearest neighbors in the positive set, while also rejecting near-duplicate patches (measured by spatial overlap of more than 30% between any 5 of their top 50 nearest neighbors). This reduces the number of candidates to about 1000.

Figure 3 (top row) shows three example candidate patches together with their nearest neighbor matches. We can see that, although some matches appear reasonable, many are not very geo-discriminative (red boxes show matches outside Paris), nor visually coherent (e.g. street sign). The

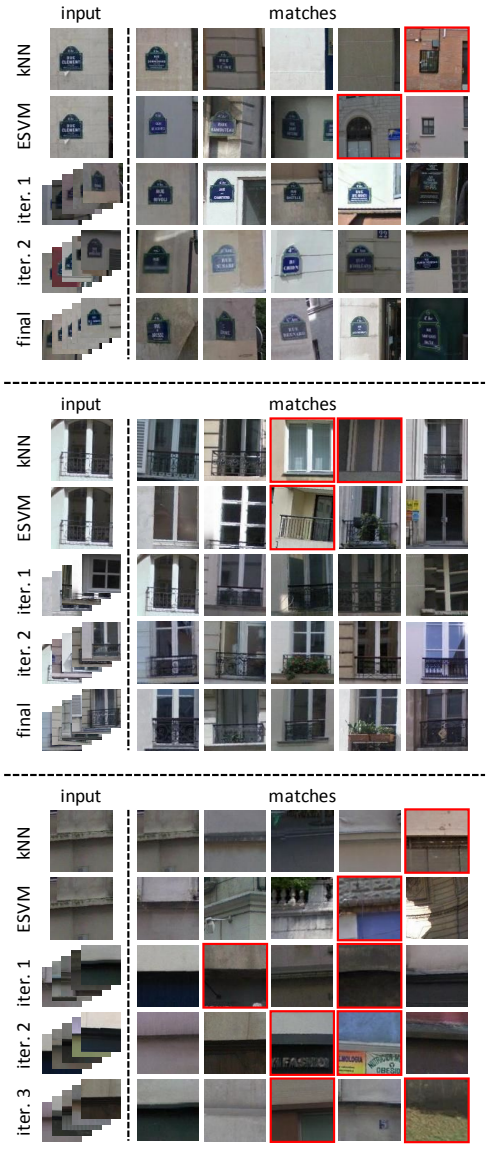

Figure 3: Steps of our algorithm for three sample candidate patches in Paris. The first row: initial candidate and its NN matches. Rows 2-4: iterations of SVM learning (trained using patches on left). Red boxes indicate matches outside Paris. Rows show every 7th match for clarity. Notice how the number of not-Paris matches decreases with each iteration, except for bottom cluster, which is eventually discarded.

main problem is that a standard distance metric, such as normalized correlation, does not capture what the important parts are within an image patch, and instead treats all pixels equally. For example, the the street sign candidate (Figure 3 center), has a dark vertical bar along the right edge, and so all the retrieved NN matches also have that bar, even though it's irrelevant to the street sign concept.

Recently, [31] showed how one can improve visual retrieval by adapting the distance metric to the given query using discriminative learning. We adopt similar machinery, training a linear SVM detector for each visual element in an iterative manner as in [33] while also adding a weak geographical constraint. The procedure produces a weight vector which corresponds to a new, per-element simi-

larity measure that aims to be geo-discriminative. Our iterative clustering works as follows. Initially, we train an SVM detector for each visual element, using the top k nearest neigbors from the positive set as positive examples, and all negative-set patches as negative examples. While this produces a small improvement (Figure 3 (row 2)), it is not enough, since the top k matches might not have been very good to begin with. So, we iterate the SVM learning, using the top k detections from previous round as positives (we set $k = 5$ for all experiments). The idea is that with each round, the top detections will become better and better, resulting in a continuously improving detector. However, doing this directly would not produce much improvement because the SVM tends to overfit to the initial positive examples [33], and will prefer them in each next round over new (and better) ones. Therefore, we apply cross-validation by dividing both the positive and the negative parts of the dataset into l equally-sized subsets (we set $l = 3$ for all experiments). At each iteration of the training, we apply the detectors trained on the previous round to a new, *unseen* subset of data to select the top k detections for retraining. In our experiments, we used three iterations, as most good clustered didn't need more to converge (i.e. stop changing). After the final iteration, we rank the resulting detectors based on their accuracy: percentage of top 50 firings that are in the positive dataset (i.e. in Paris). We return the top few hundred detectors as our geo-informative visual elements.

Figure 3 gives some intuition about the algorithm. For example, in the left column, the initial nearest neighbors contain only a few windows with railings. However, windows with railings differ more from the negative set than the windows without railings; thus the detector quickly becomes more sensitive to them as the algorithm progresses. The right-most example does not appear to improve, either in visual similarity or in geo-discriminativeness. This is because the original candidate patch was intrinsically not very geo-informative and would not make a good visual element. Such patches have a low final accuracy and are discarded.

Implementation Details: Our current implementation considers only square patches (although it would not be difficult to add other aspect ratios), and takes patches at scales ranging from 80-by-80 pixels all the way to height-of-image size. Patches are represented with standard HOG [7] (8x8x31 cells), plus a 8x8 color image in L*a*b colorspace (a and b only). Thus the resulting feature has 8x8x33 = 2112 dimentions. During iterative learning, we use a soft-margin SVM with C fixed to 0.1. The full mining computation is quite expensive; a single city requires approximately $1,800$ CPU-hours. But since the algorithm is highly parallelizable, it can be done overnight on a cluster.

4.2 Results and Validation

4.2.1 Qualitative Evaluation

Figure 4 shows the results of running our algorithm on several well-known cities. For each city, the left column shows randomly chosen images from Google Street View, while the right column shows some of the top-ranked visual element clusters that were automatically discovered (due to space limitations, a subset of elements was selected manually to show variety; see the project webpage for the full list). Note that for each city, our visual elements convey a better stylistic feel of the city than do the random images. For example, in Paris, the top-scoring elements zero-in on some of the main features that make Paris look like Paris: doors, balconies, windows with railings, street signs and special Parisian lampposts. It is also interesting to note that, on the whole, the algorithm had more trouble with American cities: it was able to discover only a few geo-informative elements, and some of them turned out to be different brands of cars, road tunnels, etc. This might be explained by the relative lack of stylistic coherence and uniqueness in American cities (with its melting pot of styles and influences), as well as the supreme reign of the automobile on American streets.

In addition to the qualitative results, we would also like to provide a more quantitative evaluation of our algorithm. While validating data-mining approaches is difficult in general, there are a few questions about our method that we can measure: 1) do the discovered visual elements correspond to an expert opinion of what visually characterizes a particular city? 2) are they indeed objectively geo-informative? 3) do users find them *subjectively* geo-informative in a visual discrimination task? and 4) can the elements be potentially useful for some practical task? To answer the first question, we consulted a respected volume on 19th century Paris architecture [22]. We found that a number of stylistic visual elements mentioned in the book correspond quite well to those discovered by our algorithm, as illustrated on Figure 6.

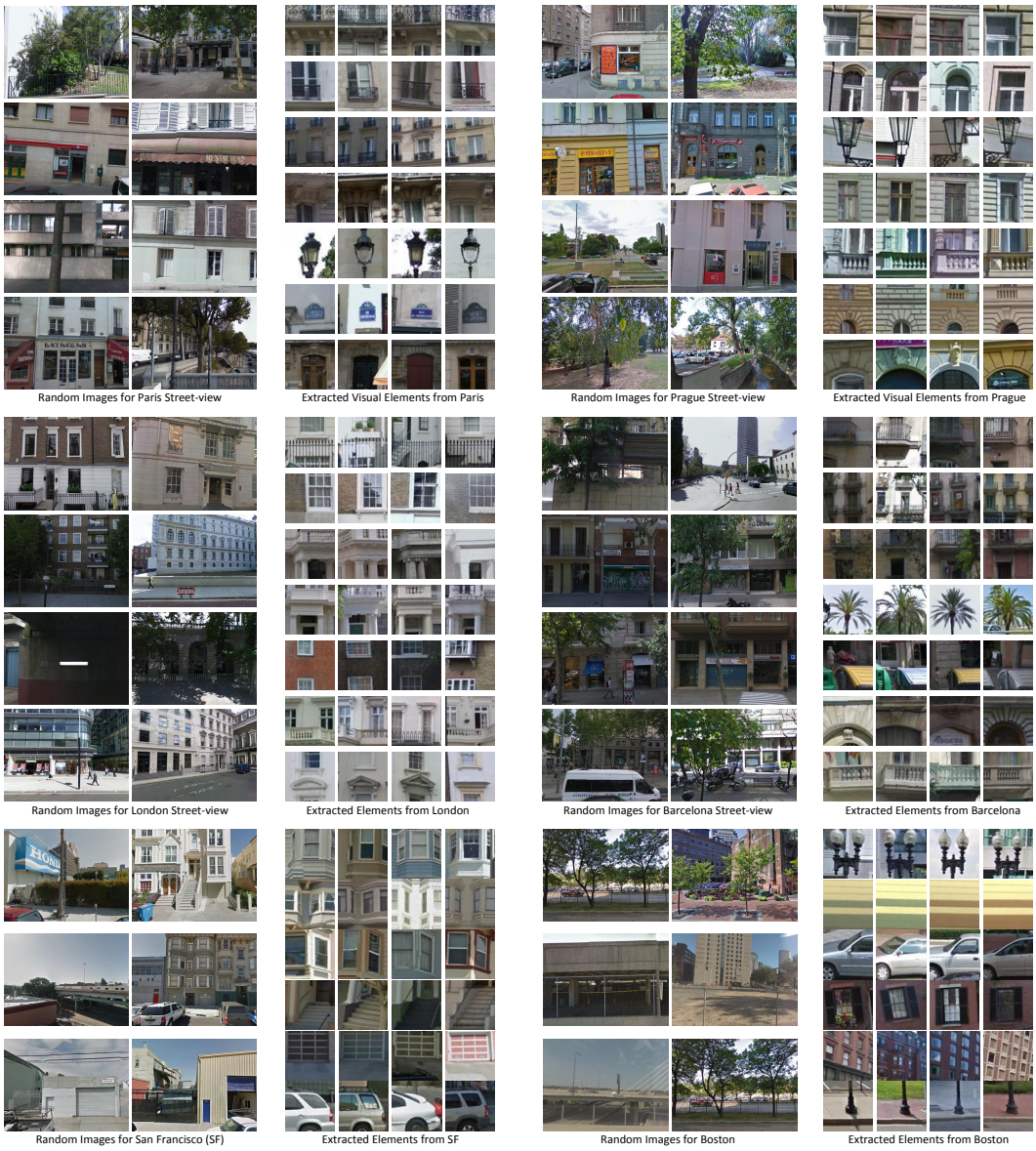

Figure 4: Google Street View vs. geo-informative elements for six cities. Arguably, the geo-informative elements (right) are able to provide better stylistic representation of a city than randomly sampled Google Street View images (left).

4.2.2 Element Accuracy

To evaluate how geo-informative our visual elements are, we ran the top 100 Paris element detectors over an unseen dataset which was 50% from Paris and 50% from elsewhere. For each element, we found its geo-informativeness by computing the percentage of the time it fired in Paris out of the top 100 firings. The average accuracy (which we term *purity*) of our top detectors was 83% (where chance is 50%). We repeated this for our top 100 Prague detectors, and found the average purity on an unseen dataset of Prague to be 92%.

4.2.3 Purity and Coverage

If a set of elements is to be useful, it should be diverse as well as accurate. To measure this, we define the coverage of an set of patches as the number of pixels contained in those patches, ignoring any

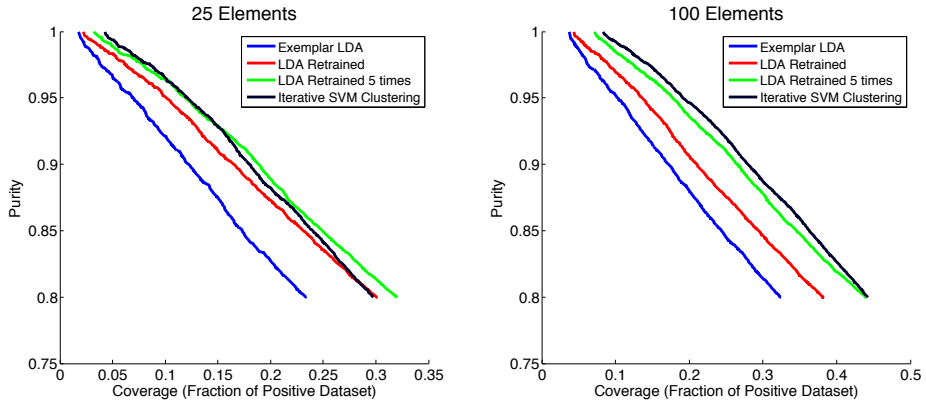

Figure 5: Purity-coverage graph for our algorithm (Iterative SVM Clustering) versus simpler baselines. In each plot, purity measures the accuracy of the element detectors, whereas coverage captures how often they fire. Higher is better.

patches from the negative set. Note that for an individual element, there is an inherent trade-off of purity and coverage: if we lower the detection threshold, we cover more pixels but also increase the likelihood of making mistakes. Hence, we can construct a purity-coverage curve for each element that is analogous to a precision-recall curve: for a given value of purity, we rank all detections for the element and select patches up until the last point in the ranking where the element has the desired purity. We then compute the coverage for this set. The purity-coverage generalizes easily to multiple elements: we simply compute the coverage of the union of all the bounding boxes associated with a given level of purity across the elements.

The purity-coverage curve is sensitive to the number of elements in a representation, since adding more elements can only increase coverage. Hence, to compare two element discovery methods, we must select equally-sized subsets of the discovered elements for each method. In this work, we only expect a few dozen elements to be truly characteristic of a city, and so we select the top 25 and then the top 100 elements for our comparison. Our purity-coverage curves are shown in Figure 5. Note that the purity-coverage curve is also sensitive to the algorithm used to select the "top" elements. To select our elements, we fix a level of purity (95%) and greedily select elements to maximize coverage on held-out data for that level of purity. This, in some sense, approximately chooses the "best" set of elements for covering the dataset. Qualitatively this produces a reasonable ranking, and more importantly, it produces similar rankings for different algorithms. However, it does tend to be biased in favor of large elements (e.g. it chooses whole facades over individual windows even if the windows are detected more accurately), but of all rankings we considered, we found it to be the most fair.

Our results are shown in Figure 5. We included three baselines of increasing complexity, and we train 20,000 elements for all of them, ultimately selecting the top ones using the above method. The simplest baseline is "Exemplar LDA," proposed in [13]. Each cluster is initialized from a single patch, and learns to separate that patch from the negative dataset, where the negative distribution is represented as a simple Gaussian. In practice, we find that the results are similar whether the negative distribution is learned from negative data or from positive data. Therefore, this algorithm is much like generating clusters via nearest neighbors (like the first steps in our algorithm), but with an improved distance metric. To show the effects of re-clustering, "LDA Retrained" takes the top 5 patches retrieved in Exemplar LDA from Paris (including the initial patch itself), and repeats LDA, separating those 5 from the negatives. This is much like the well-established method of "query expansion" for retrieval, and we can see this improves performance. Finally, "LDA Retrained 5 times" begins with elements initialized via the LDA retraining method, and re-trains the LDA classifier, each time throwing out the previous top 5 used to train the previous LDA, and selecting a new top 5 from held-out data. Hence, "LDA Retrained 5 times" is much like the iterative SVM training we propose, except that it uses LDA instead of SVMs. The iterations clearly improve performance. For the top 25 elements, our algorithm performs similarly (although slightly better in the high-putity

Figure 6: Books on Paris architecture are expressly written to give the reader a sample of the architectural elements that are specifically Parisian. We consulted one such volume [Loyer, 1988] and found that a number of their illustrative examples (left) were automatically discovered by our method (right).

regime, and slightly worse in the low-purity regime). However, for more difficult elements, our algorithm performs better. This suggests that for difficult elements, hard-mined negative data helps the algorithm learn to disciminate fine details.

4.2.4 Recognizability

Next, we examined whether the elements discovered are reliable by asking human subjects to classify them. To avoid subject fatigue, we reduced the dataset to 100 visual elements, 50 from Paris and 50 from Prague. 50% of the elements were the top-ranked ones returned by our algorithm for Paris and Prague. The other 50% were randomly sampled patches of Paris and Prague (but biased to be high-contrast, as before, to avoid empty sky patches, etc). In a web-based study, subjects (who have all been to Paris but not necessarily Prague) were asked to label each of the 100 patches as belonging to either Paris or Prague (forced choice). The results of our study (22 naive subjects) are as follows: average classification performance for the algorithm-selected patches was 78.5% ($std = 11.8$), while for random patches it was 58.1% ($std = 6.1$); the p-value for a paired-samples t-test was $< 10^{-8}$. While on random patches subjects did not do much better than chance, performance on our geo-informative elements was roughly comparable to the much simpler full-image classification task reported in the beginning of the paper (although since here we only used Prague, the setups are not quite the same).

4.2.5 Elements as Reference Art

Finally, to get a sense of whether our elements might serve as "reference art," we asked an artist to sketch a photograph of Paris, allowing only 10 minutes so that some details had to be omitted. Several days later, she made another 10-minute sketch of the same photograph, this time aided by a display of the top 10 geo-informative elements our algorithm detected in the image. In an informal, randomized survey, 10 out of our 11 naive subjects (who had all been to Paris) found the second sketch to be more Paris-like. The two sketches are shown in Figure 7.

5 Applications

Now that we have a tool for discovering geographically-informative visual elements for a given locale, we can use them to explore ways of building stylistic narratives for cities and of making visual connections between them. Here we discuss just a few such directions.

Figure 7: Geo-informative visual elements can provide subtle cues to help artists better capture the visual style of a place. We asked an artist to make a sketch from a photo of Paris (left), and then sketch it again after showing her the top discovered visual elements for this image (right). Note, for example, that the street sign and window railings are missing in the left sketch. In our informal survey, most people found the right sketch to be more Paris-like.

5.1 Mapping Patterns of Visual Elements

So far, we have shown the discovered visual elements for a given city as an ordered list of patch clusters (Figure 4). Given that we know the GPS coordinates of each patch, however, we could easily display them on a map, and then search for interesting geo-spatial patterns in the occurrences of a given visual element. Figure 8 shows the geographical locations for the top-scoring detections for each of 3 different visual elements (a sampling of detections are shown below each map), revealing interestingly non-uniform distributions. For example, it seems that balconies with cast-iron railings (left) occur predominantly on the large thoroughfares (bd Saint-Michel, bd Saint-Germain, rue de Rivoli), whereas windows with cast-iron railings (middle) appear mostly on smaller streets. The arch-supporting column (right) is a distinguishing feature of the famous Place des Vosges, yet it also appears in other parts of Paris, particularly as part of more recent Marché Saint-Germain (this is a possible example of so-called "architectural citation"). Automatically discovering such architectural patterns may be useful to both architects and urban historians.

5.2 Exploring Different Geo-spatial Scales

So far we have focused on extracting the visual elements which summarize appearance on one particular scale, that of a city. But what about visual patterns across larger regions, such as a continent, or a more specific region, such as a neighborhood? Here we demonstrate visual discovery at different geo-spatial scales.

We applied our algorithm to recover interesting patterns shared by the cities on the European subcontinent. Specifically, we used Street View images from five European cities (Barcelona, London, Milan, Paris and Prague) as the positive set, and the remaining 7 non-European cities as the negative set. Figure 9 shows some interesting discriminative features and patterns in terms of their membership across the 5 European cities. For example, while arches are common in cities across Europe, double-arches seem rare in London. Similarly, while balcony railings in Paris, Barcelona and Milan are all made of cast iron, they tend to be made of stone in London and Prague.

We also analyzed visual patterns at the scale of a city neighborhood. Specifically, we considered three well-defined districts of Paris: Louvre/Opera (1e, 2e), Le Marais (4e), and Latin Quarter/Luxembourg (5e, 6e). Figure 10 shows examples of geographically informative elements for each of the three districts (while taking the other districts and Paris suburbs as the negative set). Predictably, Louvre/Opera is differentiated from the rest of Paris by the presence of big palatial

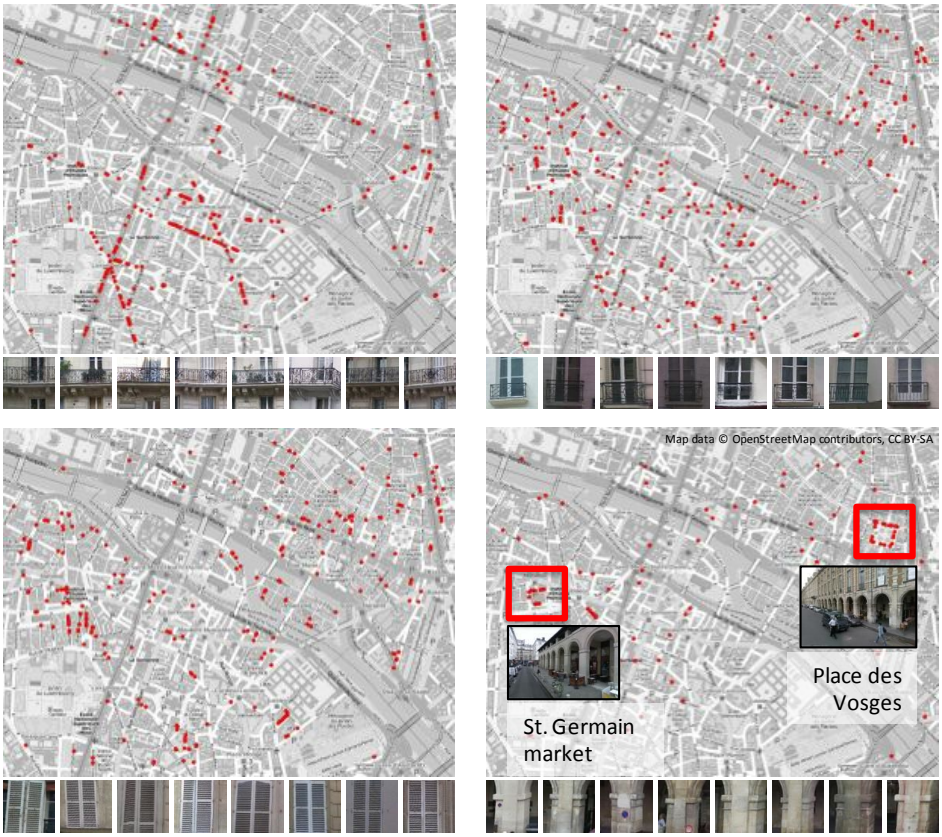

Figure 8: Examples of geographic patterns in Paris (shown as red dots on the maps) for three discovered visual elements (shown below each map). Balconies with cast-iron railings are concentrated on the main boulevards (top left). Shutterless windows with railings mostly occur on smaller streets (top right), but window shutters with slats occur on different streets (bottom left). Arch supporting columns are concentrated on Place des Vosges and the St. Germain market (right).

facades. Le Marais is distinguished by its more cozy palaces, very close-up views due to narrow streets, and a specific shape of lampposts. Interestingly, one of the defining features of the Latin Quarter/Luxembourg is the high frequency of windows with closed shutters as compared to other districts in Paris. One possible explanation is that this neighborhood has become very prestigious and a lot of its real-estate has been bought up by people who don't actually live there most of the time.

Given the detectors for visual elements at different geo-spatial scales, it becomes possible to analyze a scene in terms of the regions from which it draws its architectural influences. Figure 11 shows images from the 5th arrondissement of Paris, pointing out which elements are specific to that arrondissement, which are Paris-specific, and which are pan-European. For example, the stone balcony railings and arches are pan-European, windows with collapsible shutters and balconies with iron railings are Parisian, and the grooves around the windows are typical of the 5th arrondissement.

5.3 Visual Correspondences Across Cities

Given a set of architectural elements (windows, balconies, etc.) discovered for a particular city, it is natural to ask what these same elements might look like in other cities. As it turns out, a minor modification to our algorithm can often accomplish this task. We have observed that a detector for a location-specific architectural element will often fire on functionally similar elements in other cities,

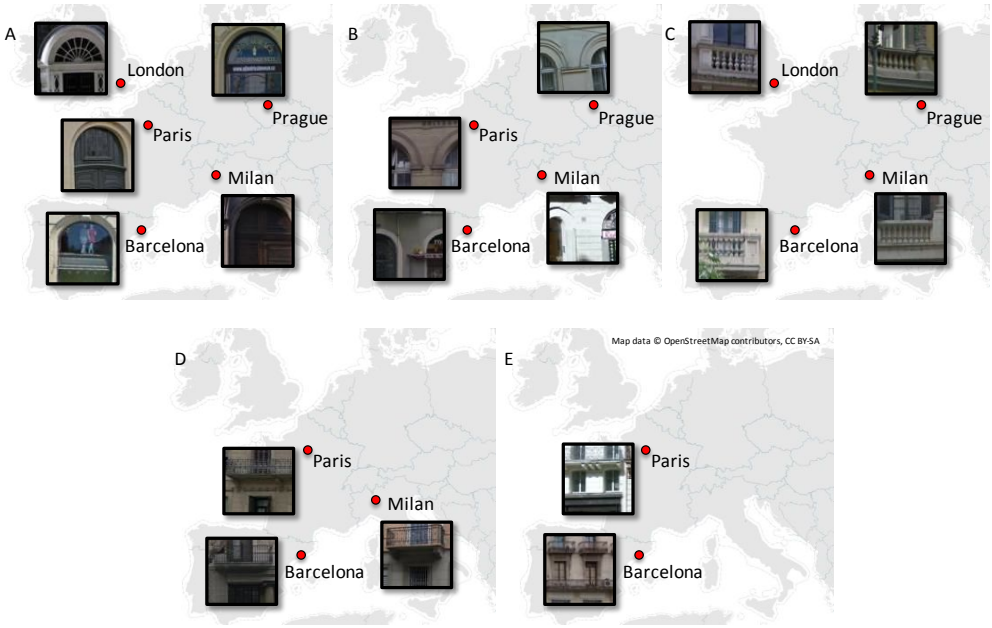

Figure 9: Architectural patterns across Europe. While arches (A) are common across all Europe, double arches (B) seem rare in London. Similarly, while Paris, Barcelona and Milan all share cast-iron railings on their balconies (D), the grid-like balcony arrangement (E) of Paris and Barcelona is missing in Milan.

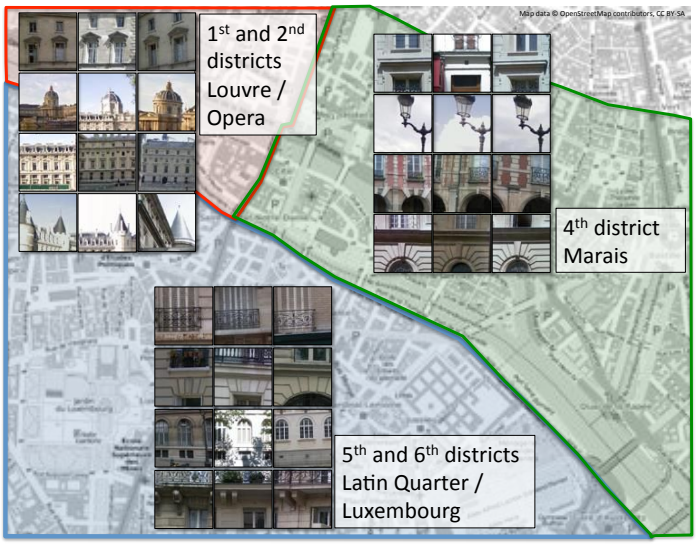

Figure 10: Geographically-informative visual elements at the scale of city neighborhoods. Here we show a few discovered elements particular to three of the central districts of Paris: Louvre/Opera, the Marais, and the Latin Quarter/Luxembourg.

just with a much lower score. That is, a Paris balcony detector will return mostly London balconies if it is forced to run only on London images. Naturally these results will be noisy, but we can clean them up using an iterative learning approach similar to the one in Section 4.1. The only difference is that we require the positive patches from each iteration of training to be taken not just from the

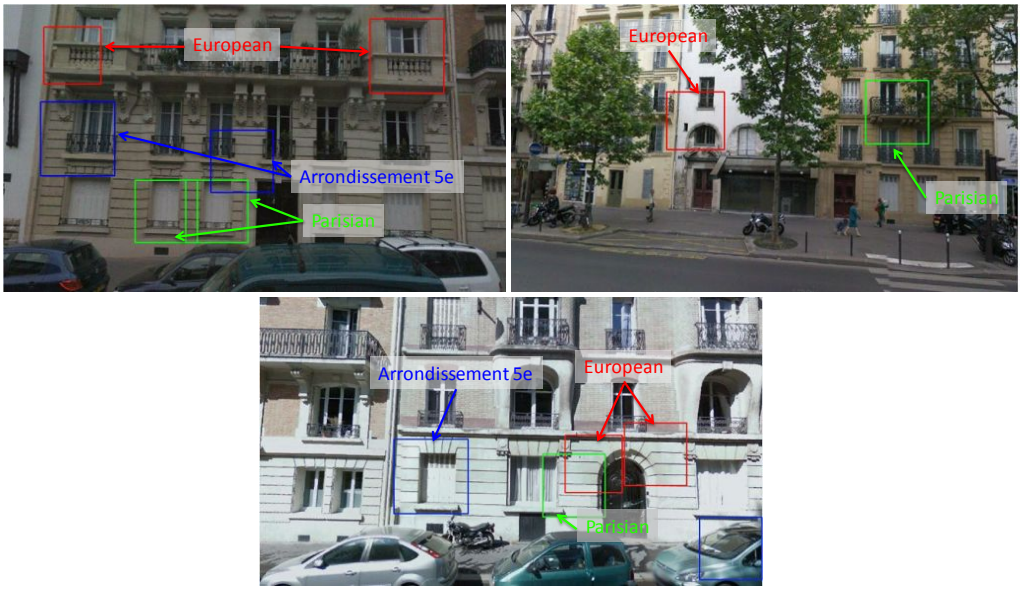

Figure 11: Detecting architectural influences. Each image shows confident detections for architectural styles at different geographic scales.

Figure 12: Visual Correspondence. Each row shows corresponding detections of a single visual element detector across three different cities.

source city, but from all the cities where we wish to find correspondences. For example, to find correspondences between Paris, Prague, and London, we initialize with visual elements discovered in Paris and then, at each round of "clean-up" training, we use 9 top positive matches to train each element SVM, 3 from each of the three cities. Figure 12 illustrates the result of this procedure. Note how capturing the correspondence between similar visual elements across cities can often highlight certain stylistic differences, such as the material for the balconies, the style of the street-lamps, or the presence and position of ledges on the facades.

Another interesting observation is that some discovered visual elements, despite having a limited spatial extent, can often encode a much larger architectural context. This becomes particularly apparent when looking at the same visual element detector applied in different cities. Figure 13

Figure 13: Object-centric image averages for the element detector in the top row of Figure 12. Note how the context captures the differences in facade styles between Paris (left) and London (right).

Figure 14: Geographically-informed retrieval. Given a query Prague image (left), we retrieve images in Paris (right).

shows object-centric averages (in the style of [37]) for the detector in the top row of Figure 12 for Paris and London. That is, for each city, the images with the top 100 detections of the element are first centered on that element and then averaged together in image space. Note that not only do the average detections (red squares) look quite different between the two cities, but the average contexts reveal quite a lot about the differences in the structure and style of facades. In Paris, one can clearly see four equal-height floors, with a balcony row on the third floor. In London, though, floor heights are uneven, with the first floor much taller and more stately.

5.4 Geographically-informed Image Retrieval

Once we have detectors that set up the correspondence between different cities such as Paris and Prague (Sec. 5.3), we can use them for geographically-informed image retrieval. Given a query image from one location, such as Prague, our task is to retrieve similar images from another location, such as Paris. For this we use the correspondence detectors from Sec. 5.3 while also encoding their spatial positions in the image. In particular, we construct a feature vector of the query image by building a spatial pyramid and max-pooling the SVM scores of the correspondence detectors in each spatial bin in the manner of [19]. Retrieval is then performed using the Euclidean distance between the feature vectors. Figure 14 demonstrates this approach where a query image from Prague retrieves images from Paris that contain similar balconies with cast iron railings (bottom) while honoring spatial layout of facades.

6 Reformulation as an Optimization Problem

While the current algorithm has good qualitative performance, it is specified in a completely procedural manner, which leaves many questions unanswered. What exactly is the algorithm optimizing? Can the iterative discarding of patches be replaced by some form or regularization that is more well understood? Can overall performance be improved? By reformulating the algorithm to have an explicit objective function, we hope to start to answer some of these questions. We also hope to address two more concrete shortcomings in the algorithm. First, for any element discovery algorithm, incorrect patches must be discarded to achieve good performance; however, the current algorithm has no good mechanism to identify bad patches, and so it discards every patch it discovers immediately after using it for training. Discarding data is inefficient from both a computational and a statistical standpoint. Second, each element does not have much opportunity to change. Specifically, the set of patches used to train the element–which completely defines the concept the SVM learns–only changes once per iteration, for a total of about 5 updates. More exploration may lead to better elements.

As before, the goal is to optimize both frequency and discriminativeness. We will again represent each element using a vector w in the feature space, and we will say that any patch with feature representation sufficiently similar to w is a member the element's cluster. We optimize w. To quantify distriminativeness, we measure the purity of the elements as the ratio of positives to negatives in a given cluster. To enforce frequency, we enforce that at least a certain number of *negatives* are included in the cluster:

$$\arg\max_{w,b} \frac{\sum_{x\in+} I(S(x,w) > b)}{\sum_{x\in-} I(S(x,w) > b)} \text{ s.t. } \sum_{x\in-} I(S(x,w) > b) = \epsilon \tag{1}$$

Here, S is a similarity function (large when w and x are similar), and ϵ is a small fixed constant, and I is the indicator function. Intuitively, enforcing that the cluster contain ϵ negatives is useful because it forces the cluster to fill up the empty regions of the space around its positive examples; otherwise, a given cluster could achieve an infinite objective function value by simply containing a single positive point. The constraint also conveniently fixes the denominator of the quotient, so we can drop it. The above function is intractable, since it is non-smooth. Hence, we relax the indicator function to a hinge:

$$\arg\max_{w,b} \sum_{x\in+} \max(S(x,w) - b, 0) \text{ s.t. } \sum_{x\in-} \max(S(x,w) - b, 0) = \epsilon \tag{2}$$

This replacement may seem somewhat arbitrary, but it actually has a strong theoretical justification, since it may actually be seen as a discriminative generalization of the classic mean-shift algorithm [9], a so-called "mode-seeking" algorithm. Mode-seeking algorithms attempt to find local maxima of a given probability distribution using only a sample from that distribution. Mode-seeking algorithms are, in general, unsupervised techniques; hence the connection in this discriminative scenario is somewhat surprising. To see the connection, let S be the negative Euclidean distance, and assume that the negative set comes from a uniform distribution. We take the limit as the size of the negative set tends to infinity, and proportionally increase ϵ, such that $\sum_{x\in-} \max(S(x,w) - \gamma, 0)/\epsilon \to 1$, for all w and some fixed γ. Since the optimization chooses b such that $\sum_{x\in-} I(S(x,w) > b) = \epsilon$, we have that $b = \gamma$, and the resulting objective function becomes simply:

$$\arg\max_{w} \sum_{x\in+} \max(S(x,w) - b, 0) \tag{3}$$

This is clearly a mode-seeking style objective. In fact, [4] showed that $\max(S(x,w) - b, 0)$ is the *shadow kernel* corresponding to the indicator function that it is replacing, and furthermore, that mean-shift moves the value of w in exactly the same direction as the gradient ascent on the above objective. Hence, in this scenario, our algorithm is equivalent to mean-shift.

How, then, do we interpret the original objective, where we have substituted the indicator function with this shadow kernel but still have a discrete negative set? It is a discriminative generalization of

Figure 15: A common problem that occurs when optimizing equation (4) directly: street numbers are a relatively rare element that look somewhat similar to street numbers in other cities. Hence, when the element discovers just a handful of classic Parisian street sign, which are much easier to separate from the negative data, it will concentrate on them, and forget about the street numbers.

mean-shift, where we are treating the negative set as a *base measure* that the positive distribution is defined over. To put it another way, we are seeking the modes of a probability density function that is defined with respect to each infinitiesimal unit volume of the negative data distribution. Hence, the actual distribution where we are performing mode-seeking can only be measured by taking the ratio of the positive distribution over the negative distribution. Note, though, that not all the theory from [4] transfers. Specifically, we can no longer say that $\max(S(x,w)-b,0)$ is a "shadow kernel," since the definition used in [4] does not have an analogue in the discriminative scenario.

Returning to the problem of visual element discovery, to follow our previous algorithm more closely (and to make the objective easier to optimize), we define $S(x,w) = w^\top x/\|x\|$. The dot product means that w still has the interpretation as a hyperplane in the feature space (rather than a hypersphere), while the normalization makes the distance behave more like a Euclidean distance. However, this means we must regularize w so that it cannot tend to infinity:

$$\arg\max_{w,b} \sum_{x\in+} \max(w^\top x - b, 0) - \lambda\|w\|^2 \text{ s.t. } \sum_{x\in-} \max(w^\top x - b, 0) = \epsilon \tag{4}$$

Note that we have not actually added a second tuning parameter here: multiplying λ by a constant and dividing ϵ by the same constant results in an identical problem. From a computational standpoint, this objective is now tractable in a datamining scenario. First, the only patches that affect the objective function's value are those for which $w^\top x - b > 0$. This means we can train our elements online, by mining for patches above or near the element's hyperplane, and discarding all others. We alternatively mine images for new cluster members, and then optimize (4) with respect to all patches mined thus far. Each such iteration of online learning parallelizes trivially in a map-reduce paradigm: specifically, we first parallelize across a set of images, mining each one for members of every cluster. New cluster members are distributed by element, such that in the reduce phase, each node is responsible for optimizing a subset of the w vectors.

In a well-behaved space, the ϵ parameter could be interpreted very much like the bandwidth parameter of a classical mode-seeking algorithm. However, for the element discovery problem, we were unable to find a single ϵ that worked well. For a rare element, a small ϵ often caused the algorithm to overfit to a single patch, whereas a large ϵ will allow the algorithm to slide off the rare element to a larger mode, usually corresponding to a more common element, as is demonstrated in Figure 15. For many elements, we have found that there is no middle ground, due to the inherent unreliability of the features we are using; hence the above objective works poorly by itself. To solve this problem, we alter the objective to treat the set of all elements as something like a global mixture model of the entire space. We then attempt to globally optimize the representativeness of our entire set of elements, through an optimization analogous to Expectation-Maximization (EM) algorithm for maximizing the likelihood of data under a mixture model. In the case of Figure 15, our hope

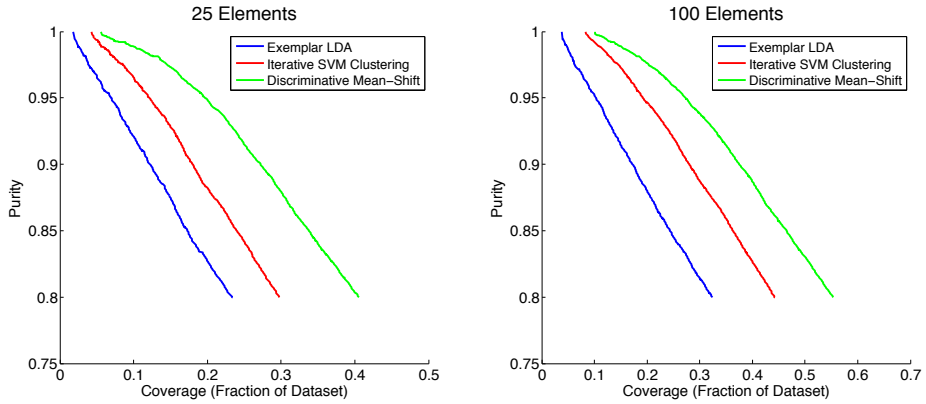

Figure 16: Purity-coverage graphs including optimization algorithm discussed in section 6. We see substantial improvement from using the our objective function.

is that there will be multiple elements that confidently detect the street sign. Hence, the handful of street signs that have made their way into the cluster depicted in the figure will be members of many element clusters, and will be well-represented by those other clusters. Hence, those element clusters containing few street signs will explore other, under-represented regions of the space.

To build our analogy with E-M, it is again useful to treat the negative data as a base measure (just like we did with mode-seeking), and to we treat the positive data as an empirical density on that base measure. This means we have a single density function to estimate, and we can approximate that density using our mixture model. Specifically, if we think of our objective function above as a "log likelihood" of the data for one component (i.e. each element corresponds to one component of the mixture), then the "likelihood" of each datapoint under the full model decomposes into a sum over components j. Like with E-M, we augment the objective function for the j'th element to introduce a weight $\alpha_{x,j}$, which measures the extent to which element cluster j "owns" or "explains" patch x. $\alpha_{x,j}$ is 1 if cluster j is the only cluster that fires on the patch x, and less than 1 if x is shared between multiple element clusters:

$$\arg\max_{w,b} \sum_{x \in +} \alpha_{x,j} \max(w^\top x - b, 0) - \lambda \|w\|^2 \text{ s.t. } \sum_{x \in -} \max(w^\top x - b, 0) = \epsilon \tag{5}$$

We alternate between solving this maximization problem (M-step) and updating the $\alpha_{x,j}$ (E-step). Unfortunately, using E-M directly to set the $\alpha_{x,j}$'s at each iteration does not work in our scenario for two reasons: (a) two similar elements will generally never have the same patch as members; rather, they will have overlapping patches as members, and (b) maximum likelihood tends to be driven by datapoints that have very low probability under the current model, whereas in our data the vast majority of datapoints are uninteresting and should not be modeled at all. To deal with (a), we compute the cluster membership of every pixel, rather than for every bounding box. Specifically, for each pixel within a bounding box, we compute the inverse of the number of bounding boxes that cover the pixel. The α for that bounding box is then the mean of these inverses. It is tempting to handle (b) as an outlier process, but in practice it is very difficult to tell the difference between outliers and good members of a cluster. Instead, we soften our E-M algorithm to act more like a mode-seeking algorithm. This is accomplished by turning off the "competition" between elements that are too similar. Specifically, during each round of the online learning, we perform agglomerative clustering on the set of element clusters, using the negative of the number of overlapping cluster members as a "distance" metric. In general, when two element clusters are representing the same underlying element, they will be clustered together. Then, as we compute the $\alpha_{x,j}$ for a given bounding box, we disregard bounding boxes for other elements in the same cluster. Hence, elements that are the same do not compete with one another, but sufficiently different elements do.

6.1 Results

Figure 16 shows the purity-coverage curves for the optimization described above ("Discriminative Mean-Shift") versus the original iterative SVM clustering and the baseline nearest-neighbors approach. We see a substantial performance gain above both methods, although the performance gain seems to be less for the more difficult elements. This is not surprising, since the optimization algorithm can make use of more than 5 instances of a particular element to learn its model. Hence, we expect the gains to be largest when there are many instances of the element to learn from. In most other respects, however, the results are quite similar; in particular, in Paris, this algorithm learned reliable detectors for all elements shown in Figure 4. This suggests that the two algorithms are indeed optimizing a similar objective, and that the optimization algorithm for the most part learns more reliable detectors for the elements previously discovered.

7 Conclusion

So, what makes Paris look like Paris? We argued that the "look and feel" of a city rests not so much on the few famous landmarks (e.g. the Eiffel Tower), but largely on a set of stylistic elements, the visual minutiae of daily urban life. We proposed a method that can automatically find a subset of such visual elements from a large dataset offered by Google Street View, and demonstrated some promising applications. This work is but a first step towards our ultimate goal of providing stylistic narratives to explore the diverse visual geographies of our world. Currently, the method is limited to discovering only local elements (image patches), so a logical next step would be trying to capture larger structures, both urban (e.g. facades), as well as natural (e.g. fields, rivers). Finally, the proposed algorithm is not limited to geographic data, and might potentially be useful for discovering stylistic elements in other weakly supervised settings, e.g. "What makes an Apple product?"

Acknowledgements: We thank Jessica Hodgins, Lavanya Sharan, and Yves Ubelmann for their valuable comments and suggestions. This work is a part of a larger effort with Dan Huttenlocher and David Crandall, on modeling geo-informative visual attributes. We thank Google for letting us publish the Street View images. This work was partially supported by NDSEG fellowship to CD and by Google, NSF IIS0905402, EIT-ICT, ONR N000141010934, ONR N000141010766, and MSR-INRIA.

References

[1] T.L. Berg and A.C. Berg. Finding iconic images. In *The 2nd Internet Vision Workshop at Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *IEEE 12th International Conference on Computer Vision (ICCV)*, pages 1365–1372, 2009.

[3] D.M. Chen, G. Baatz, K. Koser, S.S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, Xin Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 737–744, 2011.

[4] Yizong Cheng. Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790–799, 1995.

[5] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17–24, 2009.

[6] D. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg. Mapping the world's photos. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 761–770, 2009.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005.

[8] Juliet Fiss, Aseem Agarwala, and Brian Curless. Candid portrait selection from video. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 30(6):128, 2011.

[9] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, 1975.

[10] B. Fulkerson, A. Vedaldi, and S. Soatto. Localizing objects with smart dictionaries. In *European Conference on Computer Vision (ECCV)*, pages 179–192, 2008.

[11] Yunchao Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 817–824, june 2011.

[12] P. Gronat, M. Havlena, J. Sivic, and T. Pajdla. Building streetview datasets for place recognition and city reconstruction. Technical Report CTU–CMP–2011–16, Czech Tech Univ., 2011.

[13] Bharath Hariharan, Jitendra Malik, and Deva Ramanan. Discriminative decorrelation for clustering and classification. In *European Conference on Computer Vision (ECCV)*, 2012.

[14] J. Hays and A.A. Efros. Im2gps: estimating geographic information from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[15] E Kalogerakis, O. Vesselova, J. Hays, A. Efros, and A. Hertzmann. Image sequence geolocation with human travel priors. In *IEEE 12th International Conference on Computer Vision (ICCV)*, pages 253–260, 2009.

[16] Leonid Karlinsky, Michael Dinerstein, and Shimon Ullman. Unsupervised feature optimization (ufo): Simultaneous selection of multiple features with their detection parameters. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1263–1270, 2009.

[17] J. Knopp, J. Sivic, and T. Pajdla. Avoiding confusing features in place recognition. In *European Conference on Computer Vision (ECCV)*, pages 748–761, 2010.

[18] Y. Lee and K. Grauman. Foreground focus: Unsupervised learning from partially matching images. *International Journal of Computer Vision (IJCV)*, 85(2):143–166, 2009.

[19] L. Li, H. Su, E. Xing, and L. Fei-Fei. Object bank: A high-level image representation for scene classification and semantic feature sparsification. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, 2010.

[20] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *European Conference on Computer Vision (ECCV)*, pages 427–440, 2008.

[21] Y. Li, D. Crandall, and D. Huttenlocher. Landmark classification in large-scale image collections. In *IEEE 12th International Conference on Computer Vision (ICCV)*, pages 1957–1964, 2009.

[22] Francois. Loyer. *Paris nineteenth century : architecture and urbanism*. Abbeville Press, New York, 1st american edition, 1988.

[23] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *Advances in Neural Information Processing Systems (NIPS)*, volume 19, 2007.

[24] P. Mueller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *ACM Transactions on Graphics (SIGGRAPH)*, 25(3):614–623, 2006.

[25] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 155:23–36, 2006.

[26] Karen Paik. *The Art of Ratatouille*. Chronicle Books, 2006.

[27] T. Quack, B. Leibe, and L. Van Gool. World-scale mining of objects and events from community photo collections. In *Proceedings of the International Conference on Content-based Image and Video Retrieval (CIVR)*, pages 47–56, 2008.

[28] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1605–1614, 2006.

[29] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7, 2007.

[30] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[31] Abhinav Shrivastava, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Data-driven visual similarity for cross-domain image matching. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 30(6):154, 2011.

[32] Ian Simon, Noah Snavely, and Steven M. Seitz. Scene summarization for online image collections. In *IEEE 11th International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[33] Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. Unsupervised discovery of mid-level discriminative patches. 2012. `arXiv:1205.3137 [cs.CV]`.

[34] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *IEEE 9th International Conference on Computer Vision (ICCV)*, pages 1470–1477, 2003.

[35] A. Sutcliffe. *Paris: an architectural history*. Yale University Press, 1996.

[36] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios. Segmentation of building facades using procedural shape priors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3105–3112, 2010.

[37] Antonio Torralba and Aude Oliva. Statistics of natural image categories. *Network: Computation in Neural Systems*, pages 391–412, 2003.

[38] Yan-Tao Zheng, Ming Zhao, Yang Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, Tat-Seng Chua, and H. Neven. Tour the world: building a web-scale landmark recognition engine. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1085–1092, 2009.