City Forensics: Using Visual Elements to Predict Non-Visual City Attributes

Sean M. Arietta Alexei A. Efros Ravi Ramamoorthi Maneesh Agrawala

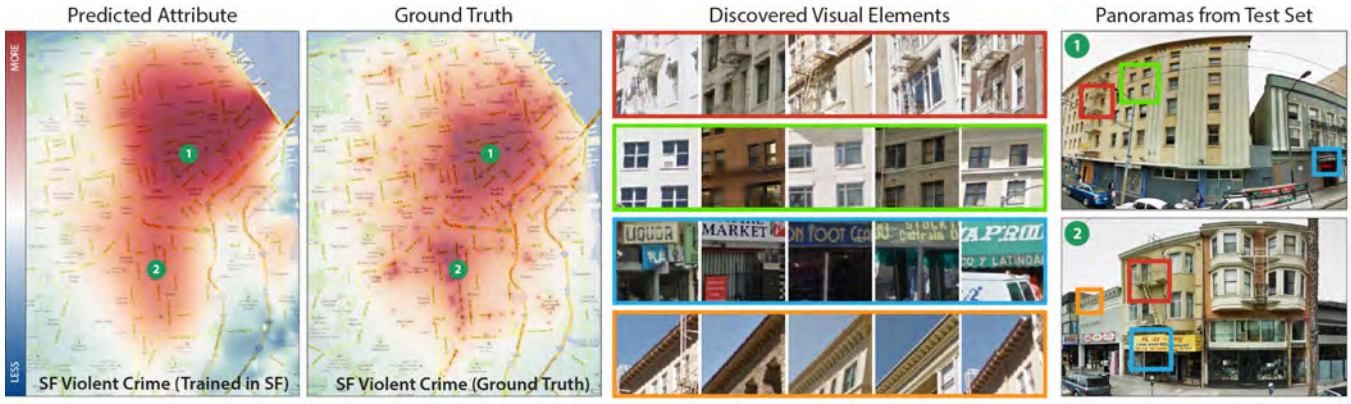


Fig. 1: The violent crime rate in San Francisco is an example of a non-visual city attribute that is likely to have a strong relationship to visual appearance. Our method automatically computes a predictor that models this relationship, allowing us to predict violent crime rates from street-level images of the city. Across the city our predictor achieves 73% accuracy compared to ground truth. (columns 1 and 2, heatmaps run from red indicating a high violent crime rate to blue indicating a low violent crime rate). Specifically, our predictor models the relationship between visual elements (column 3), including fire escapes on fronts of buildings, high-density apartment windows, dilapidated convenience store signs, and unique roof style, relate to increased violent crime rates. Our predictor also identifies street-level images from San Francisco that have an unsafe visual appearance (column 4). Detections of visual elements are outlined in color.

Abstract— We present a method for automatically identifying and validating predictive relationships between the visual appearance of a city and its non-visual attributes (e.g. crime statistics, housing prices, population density etc.). Given a set of street-level images and (location, city-attribute-value) pairs of measurements, we first identify visual elements in the images that are discriminative of the attribute. We then train a predictor by learning a set of weights over these elements using non-linear Support Vector Regression. To perform these operations efficiently, we implement a scalable distributed processing framework that speeds up the main computational bottleneck (extracting visual elements) by an order of magnitude. This speedup allows us to investigate a variety of city attributes across 6 different American cities. We find that indeed there is a predictive relationship between visual elements and a number of city attributes including violent crime rates, theft rates, housing prices, population density, tree presence, graffiti presence, and the perception of danger. We also test human performance for predicting theft based on street-level images and show that our predictor outperforms this baseline with 33% higher accuracy on average. Finally, we present three prototype applications that use our system to (1) define the visual boundary of city neighborhoods, (2) generate walking directions that avoid or seek out exposure to city attributes, and (3) validate user-specified visual elements for prediction.

Index Terms—Data mining, big data, computational geography, visual processing

1 Introduction

A modern city is a massive and ceaseless information producer, constantly generating thousands of disparate pieces of localized information (e.g. housing prices, restaurant health inspection scores, crime statistics, precipitation levels, building code violations, water usage, etc.). Since each such *city attribute* is associated with a location (latitude, longitude) we typically visualize them as attribute maps. While such maps have long been used to analyze and understand cities, urban planners have recently started applying big data analysis and mining techniques to identify meaningful correlations between these mapped attributes (so called "correlation mining") [16, 29, 46]. These correlations can then be used to *predict* the value of one attribute given another when that attribute is not readily available. For instance, higher housing prices might predict higher health inspection scores and thereby reduce the number of restaurants inspected in those areas.

However, there is one type of city data that has received far less attention from the urban planning and data mining communities – visual appearance. That is, how does the visual appearance of a city relate to its other, often non-visual, attributes? We might, for example, speculate that more trees and greenery in a neighborhood imply higher housing prices. Indeed, sociologists have proposed the Broken Windows Theory [42, 48], which suggests that visual evidence of neighborhood decay (e.g broken glass, graffiti, trash, etc.) correlates with increased levels of crime. With the widespread availability of street-level imagery (Google StreetView, Bing Streetside, etc.) we now have the data necessary to identify and validate such predictive relationships. But manually analyzing all street-level images at the scale of an entire city is impractical. Even today there has not been any large, city-scale verification of the Broken Windows Theory.

In this paper, we take a first step towards automatically identifying and validating predictive relationships between the visual appearance

- *Sean Arietta, Alexei Efros, and Maneesh Agrawala are with the EECS Department at the University of California, Berkeley. E-Mail: {sarietta,efros,maneesh}@eecs.berkeley.edu.*
- *Ravi Ramamoorthi is with the CSE Department at the University of California, San Diego. E-Mail: ravir@cs.ucsd.edu.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014; date of publication xx xxx 2014; date of current version xx xxx 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

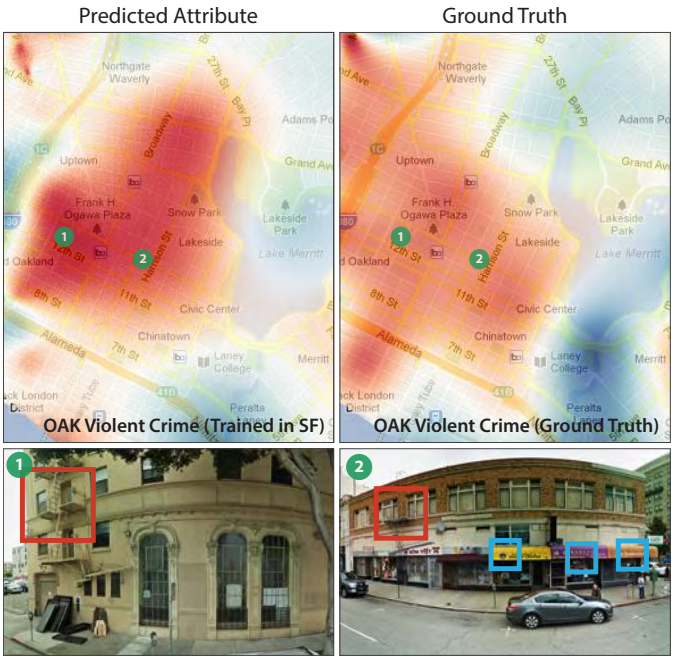


Fig. 2: Our predictor relating visual elements to violent crime rates in San Francisco (Figure 1) generalizes to predict the violent crime rate in Oakland with 64% accuracy. Our predictors can leverage this generalizability to predict city attributes in places where they may not be readily available. Street-level images (bottom row) are from Oakland, but show detections of visual elements from San Francisco (Figure 1).

of a city and its location-specific attributes. Our approach builds on the work of Doersch et al. [13] who recently introduced a method for identifying the visual elements of a city that differentiate it from other cities. However their work focuses on binary classification based only on geographic location (e.g. Paris vs. not Paris). Moreover, their method cannot directly compute the relationship between visual elements and real-valued non-visual city attributes.

Our insight is that given a non-visual city attribute and a training set of street-level images, we can extract a set of visual elements that are discriminative of the attribute. We then train a predictor that learns a set of weights over the discriminative visual elements using non-linear Support Vector Regression [43], a robust regression technique. Our predictor can then take any street-level image as input and produce an estimate of the attribute value as output. Applying the predictor across the street-level images of an entire city allows us to analyze the relationship between visual elements and non-visual city attributes, in a process we call *city forensics*.

Although certain city attributes like the presence of trees and graffiti have a natural connection to visual elements, more abstract, non-visual attributes such as crime rates and housing prices relate to visual appearance in a much more complicated, non-linear way. We show our method is indeed able to discover relationships between visual elements and some of these attributes and that this relationship is general enough to predict these attributes in new cities. For example, our system finds that for San Francisco, visual elements such as fire escapes on fronts of buildings, high-density apartment windows, dilapidated convenience store signs, and a unique roof style predict violent crime rates with 73% accuracy (Figure 1). Moreover, the predictor trained for San Francisco can also predict violent crime rates for Oakland with 64% accuracy (Figure 2). We obtain similar prediction results for a variety of other attributes including: theft rates, housing prices, population density, tree presence, graffiti presence, and the perception of danger.

To build predictors that can model the relationship between visual appearance and city attributes at the scale of a city we have developed a scalable distributed processing framework that can efficiently handle several terabytes of street-level image data. Our framework speeds up the main computational bottleneck (extracting visual elements) by an order of magnitude and requires about 3.75 hours to build a predictor for one attribute. The efficiency of our framework allows us to quickly investigate a diverse set of city attributes.

In some cases our predictions do not agree exactly with ground truth data. These discrepancies are often where we gain the most interesting insights about the connection between non-visual city attributes and visual appearance. We analyze several of these cases in Section 5. Despite these differences we show that our method enables several novel applications:

Neighborhood visual boundaries. We identify the visual boundary of neighborhoods from a set of sparse user-provided labels of areas inside and outside the neighborhood.

Attribute-sensitive wayfinding. We generate walking paths through the city that either maximize or minimize a user's exposure to a particular attribute. For example, a user might generate a path that passes through an area of high housing values.

Validating user-specified visual elements for prediction. We let users specify meaningful visual elements and then check how well those elements predict city attributes. For instance, users might test whether a visual element representing bars on windows is predictive of high crime areas.

2 Related Work

Relationships between visual appearance and non-visual city attributes have been studied in a number of fields including urban planning, sociology and epidemiology. While Broken Windows Theory [42, 48], which suggests that neighborhood decay correlates with higher crime rates, is a well-known example, others have examined how city appearance relates to attributes like obesity rates [14], cases of depression [26], and sexually transmitted disease rates [10]. Although researchers have shown that relationships between visual appearance and city attributes exist, they have had to rely on manual inspection either by the authors themselves or more recently via crowdsourcing [38, 36]. In contrast, our method relies only on a set of measurements of a city attribute and an associated set of street-level images.

Researchers have also analyzed the relationship between discriminative visual elements and the unique visual appearance of cities. For instance, Doersch et al. [13] find the distinctive visual aspects of Paris compared to London. Fouhey et al. [17] use a similar approach for computing visual elements for indoor scenes. Our approach also builds on Doersch et al. [13], but relates the visual elements to non-visual city attributes.

Several other works this year, all developed concurrently, consider the relationship between the visual appearance of cities and city attributes such as wealth, safety, beauty, quiet and proximity to specific types of landmarks [35, 34, 1, 23], as well as the relationship between city appearance and city identity [49, 3]. All these methods operate on the scale of an entire street-level image (or panorama), whereas we aim to model the visual appearance of a city at a finer, image patch scale. This gives us the advantage of being able to visualize the specific parts of the image that are most related to a given attribute.

Several techniques use image-based data sources other than street-level panoramas to detect physical attributes of cities. Aerial and street-level LIDAR has been used to detect trees [39], water and ground coverage [7], roads [6], and buildings [37]. Video is commonly used to categorize traffic in cities [24, 44] as well as to track crowds for the purposes of detecting flow patterns, anomalous behavior, etc [21, 30]. Although these techniques often produce very accurate results, the availability of these sources of image data is a limiting factor in most cities. In contrast our method relies on street-level images, one of the most ubiquitous and rapidly growing sources of data available today.

Our prediction algorithm uses non-linear Support Vector Regression [5, 43] which has been applied in a number of different fields including graphics for 3D pose estimation [2] and image restoration [28]. It has also been used to learn how visual style changes over time [27].

Fig. 3: Our predictor finds that visual elements corresponding to hedges (top), gable roofs (middle), and tropical plants (bottom) are related to high housing prices in San Francisco.

3 METHOD

Our goal is to construct a predictor that can estimate the value of a non-visual city attribute based on visual appearance. Given a set of measured (location, attribute-value) pairs and a set of (location, street-side panorama) pairs, we build predictors in three steps:

1. We spatially interpolate the input (location, attribute-value) data to obtain attribute values over the entire city.

2. We modify the technique of Doersch et al. [13] to build a bank of Support Vector Machines (SVMs) [5] that detect visual elements in the panoramas that are discriminative of the attribute.

3. We build an attribute predictor from the resulting bank of SVMs using Support Vector Regression (SVR) [43].

We consider each of these steps in turn.

3.1 Interpolating Non-Visual City Attribute Values

Although our input data consists of (location, attribute-value) pairs and (location, panorama) pairs, the locations of the attribute values may not coincide with the locations of the panoramas. But, in order to train the visual element detector SVMs and then train the attribute predictor we require a set of locations for which both the attribute-values and panoramas are available. Thus, we use radial basis functions (RBF) [31] to spatially interpolate the input set of (location, attribute-value) data. Specifically we use the inverse multiquadric function

$$f(r) = \frac{1}{\sqrt{1+(\varepsilon r)^2}} \tag{1}$$

as the radial basis function, where r is the Euclidean distance between locations and ε is a parameter that controls falloff. We have found that this basis function produces a smooth interpolation and is relatively insensitive to ε compared to other basis functions. We have found that setting $\varepsilon = 2$ works well in practice.

3.2 Constructing the Visual Element Detectors

In this work we model visual appearance using visual elements. We define a visual element as a set of image patches that have a visually consistent appearance. In Figure 3 for instance, the visual elements predictive of the housing prices city attribute in San Francisco includes rows of image patches containing hedges (top) gable roofs (middle) and tropical plants (bottom).

Doersch et al. [13] recently presented a method for constructing a bank of SVMs that can detect visual elements that are discriminative of a binary positive/negative location-based attribute (e.g. in-Paris vs. not-in-Paris) based on the method of Singh et al. [41]. To train the SVMs they require a set of locations and their associated street-level panoramas that are labeled as positives and another set labeled as negatives. To apply their method on our continuous-valued city attribute data we first threshold our attributes to form a positive set and a negative set based on the mean and standard deviation of the attribute values. For instance, if our attribute is housing price we might define positive/negative sets as locations (and corresponding panoramas) where the attribute value is greater/less than one standard deviation above/below the mean housing price.

Doersch et al. have shown that a set of about 10,000 street-level images with 2,000 positives and 8,000 negatives is sufficient to train SVMs that can detect discriminative visual elements. The positive/negative imbalance is meant to ensure there are enough negative examples to cover the larger variation of visual appearance that may occur in a negative set. To build the positive/negative sets for our continuous-valued attributes we sample the (location, attribute-value) pairs using the RBF interpolated attributes (Section 3.1) as the sampling distribution. This sampling strategy ensures that positive samples are likely to be drawn from the locations where the attribute is most positive and the same for the negative samples. We use the inverse cumulative distribution sampling method [8] to generate these samples. Finally, we associate the closest street-level panorama with each (location, attribute-value) sample. However, if the nearest panorama is more than 5 feet from the sample location we reject the sample. The result of this process is a (location, attribute-value, panorama) triplet for each sample in our training set.

To build the SVMs we must first generate image features for each of the panoramas in the training set. We extract the image features from perspective projections of Google StreetView panoramas. However, each such panorama provides a complete 360° field of view and projecting such a wide field-of view panorama into a rectangular image will introduce huge distortions. Therefore we project each panorama at 20° intervals across the entire azimuthal range from 0° to 360° and from −10° to 30° in the elevation angle also in steps of 20°. In all of our experiments we used a field of view of 30°. This gives us an overlap of 10° between successive projections maximizing the chance that every object appears whole in at least one projection. We then scale each projection to 400x400 and for each patch, over multiple scales, we extract a single image feature using the HOG+color [11] descriptor similar to Doersch et al. This produces images features of dimension 2112.

We briefly summarize the steps we perform to build the bank of SVMs for detecting discriminative visual elements from the training data (see Doersch et al. [13] for complete details).

1. Extract the set of image features in each panorama projection in the positive and negative sets.

2. Randomly sample image features from the positive set and compute their 100 nearest neighbors across both the positive and negative sets.

3. Compute an SVM for each nearest neighbor set that separates the nearest neighbor set from all other image features in the positive and negative sets.

4. Refine the SVMs using three iterations of the *hard negative mining technique* of Felzenszwalb et al. [15].

We sort the resulting SVM visual element detectors based on how well they can discriminate the positive set from the negative set and keep the top K, where $K = 100$ in our implementation. Given an image feature extracted from a panorama, each SVM produces a continuously-valued score for that feature. Positive scores indicate that the image feature is similar to the visual element that SVM is designed to detect. Note that the SVMs do not predict attribute values directly, they can only determine whether a visual element is present in an image.

3.3 Computing the Predictor

City attributes are most likely represented by a combination of many visual elements, not merely the presence or absence of a single visual element, so a direct application of the method of Doersch et al. [13] is not possible. Therefore we build our predictors by learning a set of weights over the set of scores produced by our bank of visual element SVM detectors. The simplest approach for learning these weights is to build a linear regressor between the SVM scores and the corresponding attribute values. But in practice we have found that the relationship between SVM scores and attribute values is rarely linear. Instead, we

use the more flexible technique of non-linear Support Vector Regression (SVR) [43].

In SVR, the goal is to find a function that approximates a set of training outputs y_i given a set of training input vectors x_i. In our case, given a training set of (location, attribute-value, panorama projection) triplets we treat the attribute-value as the output y_i and we treat the SVM detector scores as the input x_i. Specifically for each triplet in the training set we build a score vector for each panorama as follows:

1. We extract a set of 7,700 image features from each panorama projection (every projection is the same size and thus produces the same number of features).

2. We compute SVM scores for each of the resulting image features using the bank of 100 SVMs we constructed in Section 3.2.

3. We retain the top 3 detection scores that each of the 100 SVMs was able to achieve across all of the image features in the panorama.

This procedure leaves us with a 300-element SVM score vector for each panorama, which we treat as the input vector x_i.

SVR parameterizes the functions that approximate the y_i's given the x_i's as $f(\mathrm{x}_i) = \mathrm{w} \cdot \phi(\mathrm{x}_i) + b$, where $\phi(\mathrm{x}_i)$ is a (possibly non-linear) map on the x_i's, w is a set of weights and b is bias term. The goal of SVR is to learn the parameters (w, b) that are as *compact* as possible while minimizing the loss of a particular parameterization (w, b) in predicting the training outputs y_i. This loss is defined as:

$$L_\varepsilon(y_i, (\mathrm{x}_i, b)) = \max(|y_i - f(\mathrm{x}_i)| - \varepsilon, 0) \quad (2)$$

where ε controls the magnitude of error that we can tolerate in our fitted model. Any y_i that are within a distance ε of the fitted model are considered equally well approximated. We set $\varepsilon = 0.1$ in all of our experiments. A compact model is defined as having the fewest number of degrees of freedom in the feature space as possible. Hence, the objective in SVR is to minimize the L_2 norm of w subject to the constraints imposed by the loss function.

An important detail in SVR is the selection of the map ϕ, which transforms the input points into a higher dimensional space enabling non-linear regression. This map ϕ, is usually defined with respect to a *kernel function* $K(\mathrm{x}_i, \mathrm{x}_j)$ that operates on pairs of training inputs. We considered three different forms of K:

$$\text{Linear:} \quad K(\mathrm{x}_i, \mathrm{x}_j) = \mathrm{x}_i^T \mathrm{x}_j$$

$$\text{Polynomial:} \quad K(\mathrm{x}_i, \mathrm{x}_j) = \left(\gamma \mathrm{x}_i^T \mathrm{x}_j + r\right)^d$$

$$\text{Gaussian:} \quad K(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\gamma ||\mathrm{x}_i - \mathrm{x}_j||^2\right)$$

All of our experiments use the Gaussian kernel with γ equal to $1/D$, where D is the number of dimensions of the training vectors x_i. In our case $D = 300$ – the size of the SVM score vectors. The resulting predictor is designed to take any street-level panoramic image as input, compute its SVM score vector and then estimate the corresponding attribute value.

We use `libsvm` [9] with its default settings to apply the SVR and to compute predictions for new images.

4 Visual Processing Framework

Computing the support vector machines (SVM) via the method described in Section 3.2 requires performing a number of compute- and data-intensive operations. In our experiments the input to the algorithm is a set of 10,000 Google StreetView panorama projections (2,000 positives, 8,000 negatives), each of which is 640x640 pixels. Every image contributes 7,700 image features, each of which is comprised of 2112 floating-point numbers resulting in about 650GB of image data that we must process. In addition, the hard negative mining algorithm required to build the bank of SVMs performs several computation iterations before converging, each of which can require upwards of 26GB to be processed and produces around 2GB of new data.

To compute predictions of city attributes in new cities we apply a bank of 100 SVMs and compute detections for all of the associated visual elements in every street-level panorama projection in the city. The number of panoramas in a single city varies from 30,000 to 170,000, each of which has a resolution of 5324x2867, resulting in up to 2.5TB worth of data that must be processed.

In order to efficiently train the bank of SVMs that detect visual elements and compute predictions of city attributes across an entire city efficiently, we developed a distributed visual processing framework. We implemented our framework in C++ and we use the C Message Passing Interface (MPI) library [19] as our communication interface. This provides a considerable efficiency improvement over the implementation of Doersch et al. [13] which uses MATLAB and relies on a shared filesystem for communication. Our framework assumes that all inputs and outputs are expressed as matrices and that the rows/columns of an output matrix are updated only once. These assumptions enable a number of implementation features:

Scheduling. Our restriction that all variables must be expressed as matrices allows us to split inputs by columns or rows and iteratively process them on different compute nodes in parallel. When the output of a node is available, the system updates the output matrix accordingly and requeues the node with a new set of input rows/columns. By executing jobs iteratively in this manner, slower nodes receive fewer rows/columns and inefficient or malfunctioning nodes can be removed altogether. Thus, the total execution time of a job is not bound by the slowest node.

Checkpointing. Output matrices that are partially completed are periodically saved to disk and can be reloaded if a failed job is restarted.

Caching. Variables that need to be repeatedly sent to processing nodes within a single job and across jobs can be flagged to be cached on the nodes' local filesystems. For large data that remains the same across jobs or across a single job (e.g. a list of all of the images being processed with their associated metadata), the framework can instruct the nodes to load the variables from their local cache rather than re-transferring them.

Partial variables. The assumption that the rows/columns of an output matrix are updated only once enables us to significantly reduce the memory footprint of our jobs. We save partially completed blocks of large output matrices to disk as they finish, which avoids having to keep them in memory.

We have deployed our framework on a local heterogeneous 38 core cluster, the Texas Advanced Computing Center supercomputers Lonestar and Stampede, and on Amazon's Elastic Compute Cloud. With our efficiency features, our framework can compute the SVMs via the visual element extraction technique (Section 3.2) for a single city attribute in about 3.75 hours on the Stampede supercomputer using 48 effective cores (181 CPU hours). This is a 9.9x speedup compared Doersch et al. [13]. This speedup allows us to apply our city forensics data-mining approach to analyze a diverse set of city attributes and check for correlations with visual appearance. We can compute predictions over an entire city of 170,000 panoramas (the largest set we have) in 272 CPU hours, or about 1.7 hours on the Stampede supercomputer using 160 effective cores. Our framework has been released and is available online at `http://github.com/ucb-silicon/cesium`.

5 Results

We analyzed the performance of our city attribute predictors in estimating the value of the following city attributes in 6 American cities where available:

	San Francisco	Chicago	Boston	Oakland	Seattle	Philadelphia
San Francisco	0.712	0.495	0.493	0.631	0.615	0.592
Chicago	0.584	0.647	0.577	0.571	0.545	0.650
Boston	0.599	0.542	0.815	0.472	0.444	0.722

Fig. 4: The ROC curves and the area under those curves for the housing price attribute show an interesting phenomenon. Although Boston and Philadelphia have a similar visual appearance related to housing prices, San Francisco does not generalize well to these cities.

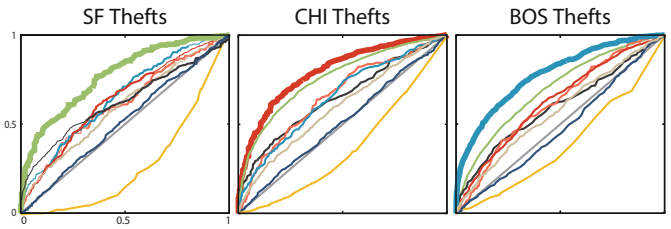
- ***Violent crime rate.*** We used the past year's worth of data from CrimeMapping.com or CrimeReports.com (depending on availability) of occurrences of assault, homicide, or robbery. We also consider the ***violent crime rate normalized by population density***.
- ***Theft rate.*** We used the past year's worth of data from CrimeMapping.com and CrimeReports.com of occurrences of theft (does not include robbery). We also consider the ***theft rate normalized by population density***.
- ***Housing prices.*** We used prices of homes sold in the past three years according to the online real estate company Zillow.
- ***Population density.*** We used population estimates from the 2010 Census normalized by the area of the associated Census Block.
- ***Tree presence.*** We used locations where people noted trees larger than 70 centimeters via the website UrbanForestMap.org, which was only available for San Francisco.
- ***Graffiti presence.*** We used reports of graffiti submitted to the 311 websites of San Francisco, Chicago, and Boston.
- ***Perception of danger.*** We deployed a Mechanical Turk experiment asking workers to examine a series of 15 Google StreetView panoramas from San Francisco, Chicago, or Boston and "decide based on the image alone whether [they] would feel safe in the area or not at any time of day." We tested a total of about 500 panoramas per city. For each image we averaged the responses of 10 workers to determine the attribute value.

Many of these sources do not provide real-valued attribute data. Crime data, for example, is only available as individual occurrences of theft, robbery, homicide, etc. To convert these discrete measurements into continuous rates we use the interpolation scheme described in Section 3.1. This step is necessary for violent crime, theft, tree presence, and graffiti presence.

In all cases where data was available we trained predictors in San Francisco, Chicago, and Boston and tested the predictors in San Francisco, Chicago, Boston, Oakland, Philadelphia, and Seattle.

5.1 Analysis of Prediction Accuracy

Figures 4–6 present quantitative accuracy results for our attribute predictors. Each entry in the tables represents the area under the receiver operating characteristic (ROC) curve for a predictor trained in one city (rows) and tested in another city (columns). Additional ROC curves and their associated tables are available in the supplemental materials. Note that when the training and test cities are the same we ensure that the set of (location, attribute-value, panorama) triplets used for testing are completely disjoint from the sets of triplets used to train the visual element detectors (Section 3.2) as well as the sets of triplets used in the SVR training (Section 3.3).



	San Francisco	Chicago	Boston	Oakland	Seattle	Philadelphia
San Francisco	0.772	0.639	0.653	0.625	0.616	0.597
Chicago	0.760	0.798	0.652	0.660	0.650	0.599
Boston	0.694	0.629	0.779	0.621	0.598	0.575
Mech. Turk	0.264	0.368	0.383			
	Normalized by Population					
San Francisco	0.892	0.772	0.658	0.651	0.628	0.682
Chicago	0.760	0.812	0.635	0.595	0.642	0.658
Boston	0.696	0.671	0.618	0.565	0.586	0.620

Fig. 5: For the theft rate attribute we compare the accuracy of our predictors to human predictions obtained via Mechanical Turk. We showed workers panoramas of San Francisco, Chicago, and Boston and asked whether they "would feel safe at all times of the day". On average our theft rate predictors are about 33% more accurate than humans. We also compare to population-normalized theft rates, which perform 5%-10% better in most cases. The increase in accuracy is a result of factoring out the correlation between theft rates and population density.

Violent Crime

	San Francisco	Chicago	Boston	Oakland	Seattle	Philadelphia
San Francisco	0.734	0.608	0.580	0.644	0.634	0.519
Chicago	0.650	0.636	0.616	0.591	0.604	0.596
Boston	0.552	0.563	0.662	0.539	0.556	0.574

Violent Crime (Normalized by Population)

	San Francisco	Chicago	Boston	Oakland	Seattle	Philadelphia
San Francisco	0.893	0.700	0.653	0.611	0.721	0.593
Chicago	0.722	0.712	0.607	0.573	0.640	0.584
Boston	0.644	0.605	0.602	0.611	0.552	0.588

Population Density

	San Francisco	Chicago	Boston	Oakland	Seattle	Philadelphia
San Francisco	0.896	0.668	0.689	0.645	0.604	0.819
Chicago	0.644	0.767	0.600	0.584	0.586	0.658
Boston	0.684	0.607	0.762	0.602	0.567	0.745

	Graffiti			Perception of Danger		
	San Francisco	Chicago	Boston	San Francisco	Chicago	Boston
San Francisco	0.682	0.553	0.615	0.748	0.526	0.589
Chicago	0.482	0.559	0.550	0.513	0.823	0.629
Boston	0.569	0.496	0.686	0.570	0.602	0.756

Fig. 6: Each table contains the area under the ROC curves for one of the city attributes we considered. Cities used to train predictors are noted in the first column and testing cities are noted in the first row. In most cases our predictors are more accurate than random classification (0.5).

The ROC curve is defined as the relationship between the true positive rate (TPR) and the false positive rate (FPR) of a binary classifier. A value of 0.5 indicates that the classifier is not discriminative at all i.e. its accuracy is equivalent to that of a random classifier, regardless of the prior distribution on positives and negatives. Anything between 0.5 the maximum area of 1.0 is considered discriminative [20]. To compute the area under the ROC curves for our predictors we first build ground truth estimates for each of the attributes the same way we generate the positive and negative sets when we construct visual element detectors (Section 3.2). We convert our real-valued attribute predictions into binary classifications and compare to ground truth while varying a positive/negative cutoff threshold. To produce the curves we varied this threshold from 0 to 1 and computed the TPR versus FPR for each threshold value.

For most of the city attributes we tested, the area under the ROC curves for the intra-city predictors (diagonals in tables) was above 63% with several of the predictors achieving above 77% accuracy. The

Fig. 7: The visual elements that are predictive of graffiti in Chicago do not contain actual examples of graffiti. Since we train our predictor on reports of graffiti, our predictors are modeling the relationship between visual appearance and the likelihood that graffiti will be present, not graffiti itself.

best result was for housing prices in Boston, which was 82% accurate (Figure 4). The only intra-city predictor to perform worse than 63% was the graffiti predictor for Chicago which attained only 56% accuracy. Although there is a strong relationship between visual appearance and actual graffiti, the training data we provided to the system does not necessarily contain examples of actual graffiti –it only contains locations where graffiti has been reported. Figure 7 shows the visual elements for the graffiti predictor trained in Chicago. None of these visual elements contain examples of graffiti. Rather the visual elements capture the appearance of areas where graffiti is likely to be observed (e.g. brick walls, window styles associated with specific neighborhoods, and street lights). Instead we are learning a relationship between visual appearance and locations where graffiti is likely to be found, and we find that this relationship not very predictive. That is, in Chicago graffiti occurs in many visually diverse places.

While the results of the cross-city predictions show a range of performance, more than half of the predictors are more than 60% accurate. Some notable results are the San Francisco population density predictor in Philadelphia (82%), the Chicago theft predictor in San Francisco (76%), and the Boston housing price predictor in Philadelphia (72%). The last case (Figure 4) illustrates an interesting phenomenon. Boston and Philadelphia are spatially close and share a similar architectural history, so it is not surprising that a housing prices predictor trained in Boston performs well in Philadelphia. However, the accuracy of the San Francisco housing prices predictor is relatively low in Boston and Philadelphia, having approximately the same accuracy as a random predictor. This result also makes sense considering the large differences in their respective histories and architectural styles. In fact, brick buildings, which are indicative of high housing prices in Philadelphia and Boston (Figure 8), are no longer allowed to be built in San Francisco due to earthquake concerns. Visual elements for all the attributes we analyzed are available in the supplemental material.

Normalized versus Non-Normalized Crime Rate Predictors. Normalized crime rate predictors relate visual appearance to the likelihood that an *individual* will be a victim of crime. In contrast, non-normalized crime rate predictors relate visual appearance to the likelihood of a crime occurring. Thus, without normalization, the resulting predictors are correlated with population density; there are more crimes where more people live. When we factor out the population density for violent crime rates and theft rates, we see a 5%-10% increase in the accuracy of our predictors in almost all cities (Figures 5, 6).

Comparisons to Human Prediction Accuracy. We compare the accuracy of our theft rate predictors to human performance using the data we collected using Mechanical Turk on the perception of danger. As shown in Figure 5 we compare the human danger ratings to the ground truth for theft rates to produce the yellow ROC curves. For San Francisco, Chicago, and Boston Mechanical Turk workers could predict theft rates with 26%–38% accuracy, whereas our predictors are 63%-80% accurate in these cities. On average our predictors outperform humans by 33%.

One potential reason for the low accuracies of some of our predictors is that ground truth attribute data may be incorrect. For example data collected about the presence of trees may be out of date. To assess such inconsistencies, we evaluated the performance of our predictions on the presence of trees in San Francisco with respect to our ground truth data and to data generated by humans. We asked Mechanical Turk workers and Facebook friends to decide whether a set of street-level images contained "at least 50% of a single tree". We tested 1000 images of San Francisco and an average of 10 people marked each image. Our prediction accuracy compared to our ground truth was 75%, but increased to 81% when we considered the human-derived ground truth. (See supplemental materials for a comparison of the ROC curves.) Unfortunately determining ground truth for non-visual city attributes is a difficult process as humans are often inaccurate at predicting such attributes from visual appearance as we discovered in the comparison of our theft predictor to human perception of danger (Figure 5).

5.2 Prediction Maps

Figures 1, 2 and 8 show how we can use our predictors to generate predicted attribute maps for an entire city. The first column shows our interpolated predictions using a predictor trained in the city noted in parentheses. The second column shows a map of the ground truth attribute values for the test city. We show some of the visual elements that are related to the attributes in column 3. The final column shows some example panoramas from the testing cities corresponding to locations denoted in our prediction maps in column 1.

While our predicted attribute maps do not always agree with the interpolated ground truth maps, there is significant overlap. Some of the inconsistencies between the maps are a by-product of the assumption that visual appearance is always related to non-visual attributes. For instance, in the Philadelphia housing price maps (second row in Figure 8) there is a large section near the 2 marker that we have predicted to have low housing prices, but ground truth indicates this is an area of high housing prices. Indeed the corresponding panorama does not have the visual appearance of a high housing price area. The discrepancy between our predictions and ground truth likely occurs because there is not a clear link between visual appearance and housing prices in this case.

Another interesting example we found is for theft rates in Chicago (top row). Although both of the panoramas from the marked areas exhibit a relatively non-threatening appearance and are from shopping districts in downtown Chicago, our San Francisco predictor indicates the theft rate is high. While at first these panoramas do not seem to have the visual appearance of high theft rate areas, the ground truth data indicates that our predictions are correct.

5.3 Deep Convolutional Neural Network Features

Recent work has shown that classification and recognition algorithms that use features extracted from a deep convolutional neural network can outperform HOG features [40, 18]. We have conducted initial experiments replacing the HOG+color features described in Section 3 with deep convolutional features generated using Caffe [22]. Specifically, we used the fifth convolutional layer of the Caffe ImageNet model as our features. Each such feature contains 2304 dimensions. The rest of the algorithm — multi-scale sliding window feature extraction, SVM training, etc — was kept exactly the same.

Figure 9 shows that the Caffe features produce visual elements that capture higher-level aspects of visual appearance, but are less visually consistent than those generated using HOG+color features. For instance, the image patches for the first visual element (top row) for San Francisco violent crime rate (normalized by population) are all instances of signs but vary in color, orientation, and font. Likewise, the patches for the second visual element (bottom row) for San Francisco housing prices contain white buildings with tall slender windows in various orientations, but sometimes contain greenery. Finally, the first visual element (top row) for reports of graffiti in San Francisco is comprised of textured image patches some of which are examples of graffiti (patches 4, 5, 7). As shown in Figures 1, 3, 7, 8, 10, 11, 12,

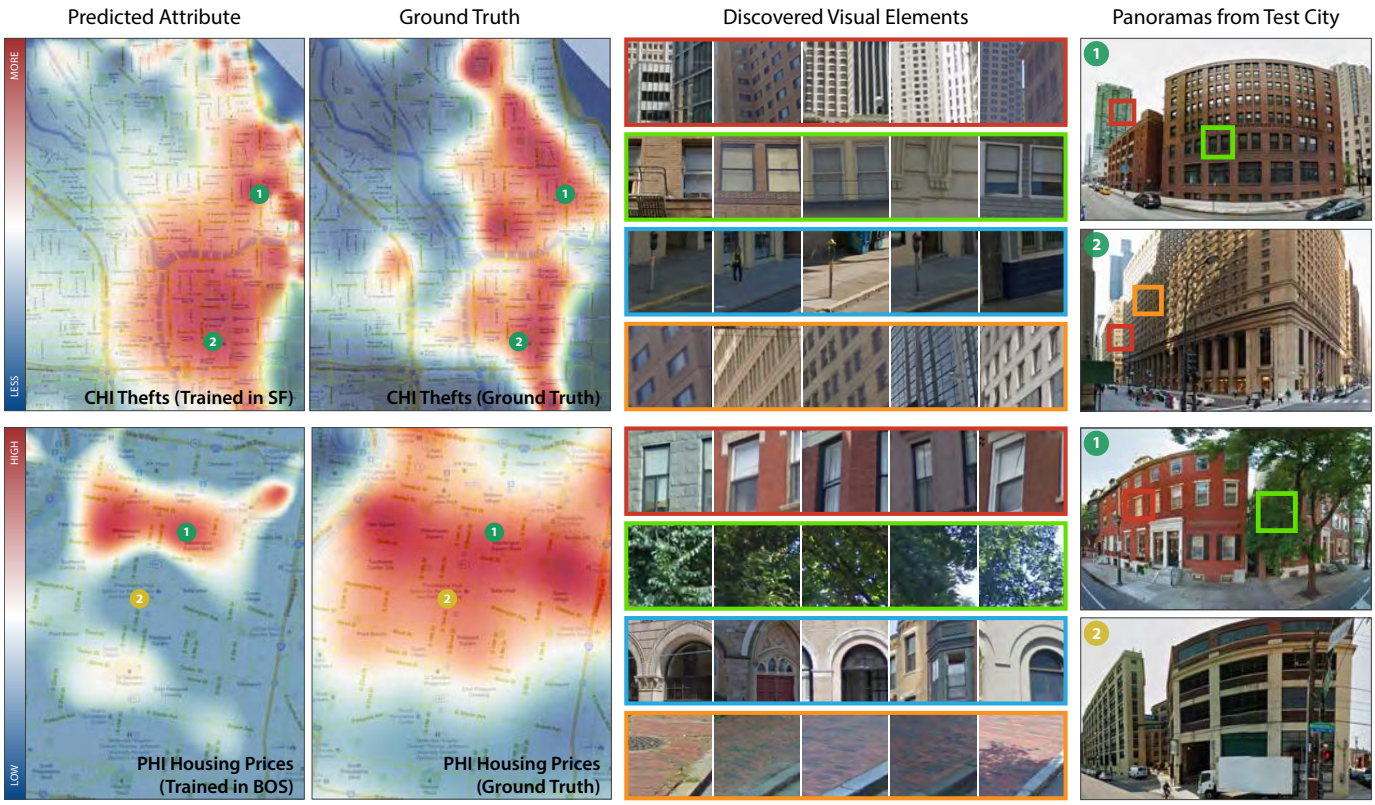


Fig. 8: Our predictors are general enough to predict the value of city attributes in test cities using predictors computed in training cities. Our method closely matches the ground truth attribute values in the test cities in most cases. Our predictors detect a set of visual elements in each panorama in the test cities and combine the detection scores of these elements into a single prediction of the attribute value. Errors can occur when the value of a city attribute is not directly related to visual appearance as in the case of housing prices in Philadelphia (second row). In this case the panorama in Philadelphia at location 2 does not have the visual appearance of a high housing price area, but the ground truth indicates that it is.

the visual elements generated using HOG+color features are more visually consistent, but capture less of the semantics of city appearance.

Quantitatively comparing the accuracy of the Caffe features to HOG+color features we obtain mixed results. For instance, while the San Francisco housing prices predictor trained with Caffe features is 14% better (63.8% versus 49.5%) in Chicago, the graffiti predictor with Caffe features predicts reports of graffiti in Chicago with 9.6% less accuracy (47.4% versus 56.9%). In ongoing work we are exploring techniques to combine the advantages of both types of features to take advantage of the strengths of each one.

6 Applications

We have implemented three applications that use our predictors to provide estimates of city attributes.

6.1 Defining Visual Boundaries of Neighborhoods

City neighborhoods often have a unique visual appearance. Although city governments impose their own definitions of the boundaries of neighborhoods, understanding the visual boundary of a neighborhood is often useful for tourists, marketers, prospective home buyers, etc. We developed a prototype application that can determine the visual boundary of neighborhoods from a sparse set of user inputs.

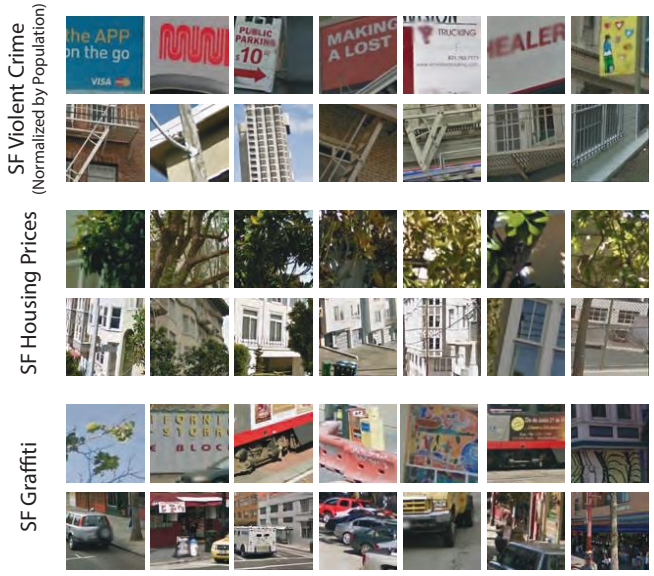
We require a user to "paint" regions of a city defining areas that are definitely part of a neighborhood and areas that are definitely not. For instance, in Figure 10 a user has painted an area of San Francisco inside the Mission neighborhood (red) and an area outside the Mission (blue). We use the paint strokes to define the positive and negative sets that are used to train our predictors. To determine the visual boundary of the neighborhood, we compute predictions on every panorama between the two user-defined paint strokes (first column in Figure 10). Note that the visual elements used by the predictor (third column in Figure 10) capture the unique visual appearance of the Mission including, Victorian homes with bayview-style windows, metal gates and bars on the fronts of homes, dilapidated signs, and bike lanes.

Our boundaries show better agreement with the actual appearance of the Mission neighborhood than the "ground truth" boundary, which we computed as the union of the boundaries defined by the San Francisco local government, realtor association, and from the analysis of Wahl and Wilde [47]. For instance, in the region marked 1 in both maps, the corresponding panorama (top right) looks much more industrial than the classic Victorian/Edwardian look of the Mission (bottom right). Our prediction map correctly leaves the location outside the boundary of the Mission, whereas the traditional ground truth neighborhood map includes it.

Personalized Attribute Explorer. Although in this example we showed a user specifying regions that define a neighborhood, this idea can be generalized. Given a set of user-defined locations in a city, we can generate a personalized attribute predictor that would enable the user to find other locations with a similar visual appearance. We believe a system like this would be useful for exploring new areas of cities without forcing users to verbally define exactly what they find visually interesting.

6.2 Attribute-Sensitive Wayfinding

Traditional navigation systems compute routes between two user-specified locations in a city that minimize the total distance traveled. But specific user groups such as tourists, urban hikers, and consumers often may want to compute routes between locations in a city that avoid or seek out certain city attributes, rather than finding the shortest/fastest route. For instance, a tourist who is unfamiliar with the downtown area of Chicago might want to avoid areas of high theft rates while walking between two locations in the area even if that means



	San Francisco	Chicago	Boston
SF Violent Crime	0.891	0.685	0.690
SF Housing Prices	0.695	0.638	0.573
SF Graffiti	0.608	0.474	0.622

Fig. 9: When we use deep convolutional network features instead of HOG+color features the resulting visual elements are able to capture more of the semantics of city appearance, but are less visually consistent. In addition, although the new features provide a slight increase in accuracy in some cases (e.g. housing prices in Boston), we also observe large decreases (e.g. violent crime predictions in San Francisco).

walking for a slightly longer period of time.

Given a user-specified city attribute and a city to compute routes in, we first model the city's connectivity as a set of nodes in a graph, where each node corresponds to a single street-level panorama in the city and is connected to all other nodes that can be reached via the actual road network. We treat each prediction of the city attribute as a weight on the corresponding node and use Dijkstra's shortest path algorithm [12] to compute the attribute-sensitive route in realtime as the user adjusts the importance of avoiding or seeking out the attribute.

In Figure 11 we show an example of our system being used to generate a walking route that avoids high theft rates in downtown Chicago. A typical route (green) simply optimizes for the total travel distance, but passes through an area where a large number of thefts have been reported (red circles in second column). Our predictions correctly avoid this path and instead route the user through a lower theft rate area. In this case, we used a predictor of theft rates trained in San Francisco to compute the predictions, highlighting the novelty of our predictors' ability to generalize to cities not used during training.

In the last two columns of Figure 11 we see that the predictor is avoiding the green path because it passes through an area with high-density buildings and areas where people are likely to be passing through quickly (parking meters indicate no long-term visits). In contrast, our theft-avoiding route passes through less dense areas of the downtown area where there are parks and wide-open spaces; places where successfully committing a theft is less practical.

It is possible to combine this approach with the *Personalized Attribute Explorer* discussed at the end of Section 6.1 to create a system for finding routes that pass through visually interesting areas as defined by a user. For instance, a user might paint strokes in areas that he/she enjoys walking through, without having to specify directly what about those areas is visually interesting. If there is a consistent visual appearance in those areas, the resulting predictor could be used to find other routes in the same city or routes in new cities that expose him/her to the same visual appearance.

6.3 Validating Visual Elements for Prediction

Despite the numerous ways researchers have manually analyzed the relationships between city attributes and visual appearance [10, 14, 26, 36, 38, 48], there is currently no automatic way to answer simple questions like: "Does (blank) discriminate the city attribute (blank)?". For instance, the top row of Figure 8 indicates that things like high-density windows, drab windows, and parking meters are predictive of theft, but our initial intuition was that visual elements like "bars on windows" would be more predictive.

In the top left column of Figure 12 we show 3 images containing examples of bars on windows and 3 images containing examples of high-density windows that we hand-picked from Google StreetView images of San Francisco. We modified our system to use patches from these images as initial seeds for the visual elements rather than randomly sampling patches from images of high theft-rate locations. Apart from this modification, the predictor was trained with respect to the theft rate attribute in San Francisco using the same method described in Section 3.

To determine whether a user-defined visual element is discriminative of an attribute two conditions must be met: (1) the predictor must converge on visual elements that detect patches which are visually consistent with the seed set and (2) a high percentage of the detected patches must come from the positive test set. Figure 12 (middle column) shows the top 3 resulting visual elements for both sets of seed images. In both cases the visual elements satisfy condition (1); the detections of the visual elements have a consistent visual appearance with the seeds images. The plot on the far right of Figure 12 shows the percentage of detections of each visual element that are from the positive set (i.e. from panoramas where a theft occurred). Many of the top detections of the "bars on windows" visual elements are from the negative set, which violates condition (2). Since the plots for the "high-density windows" visual elements show a consistently higher percentage of detections from the positive set, they are more predictive of theft rates than the "bars on windows" visual elements.

This allows us to interactively verify our intuitions about what is visually predictive of city attributes in a straightforward way. It also has the potential to assist researchers in validating their work. Efforts like the PlacePulse study [38] could use this system to validate whether human perception of safety is indeed visually predictive of actual safety by providing examples to our application of perceived safe locations.

7 Conclusions and Future Work

We present a method for automatically computing predictors of non-visual city attributes based on visual appearance. We model visual appearance using visual elements that are discriminative of the city attributes. We show that there is indeed a predictive relationship between visual appearance and non-visual city attributes in a number of cases. We also present three applications that use our predictors to provide estimates of city attributes, demonstrating the applicability of this work.

We imagine our predictors being used to perform what we call *city forensics*. That is, we envision users investigating abstract non-visual city attributes in more intuitive visual ways via the predictive relationships we are able to uncover. We also believe that our predictors could be used to synthesize new content that has the visual appearance of city attributes, which would be useful in areas like procedural city generation [32, 33] and city simulation [4, 45].

Acknowledgments

This work was supported in part by the Intel Science and Technology Center for Visual Computing, NavTeQ, a Google Research Award, and computing resources from XSEDE (via NSF grant 1011832), supported by NSF grant number ASC-130027. Additional equipment was provided by Nokia and Samsung.

References

[1] Aesthetic capital: What makes london look beautiful, quiet, and happy? In *Proceedings of CSCW 2014*, 2014.

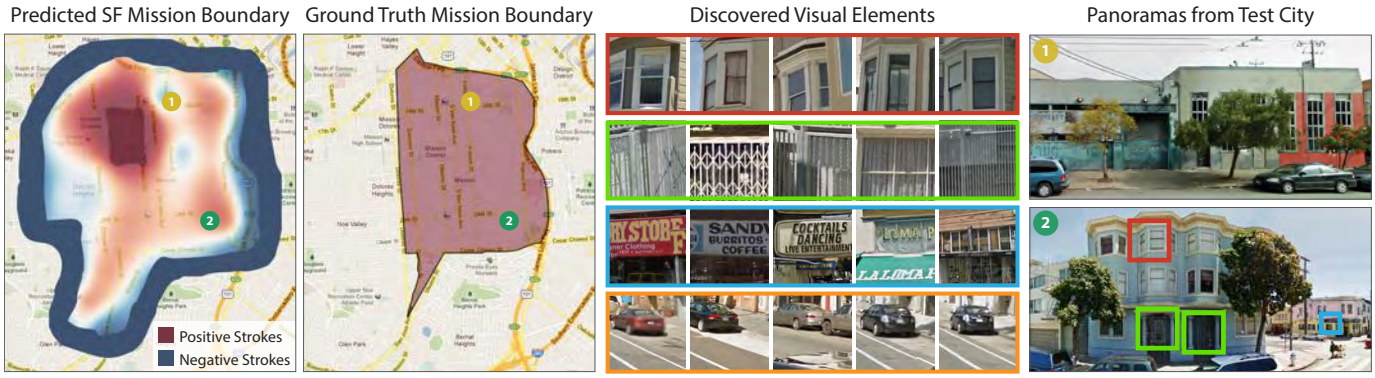


Fig. 10: Given a user-specified set of regions that are definitely part of the Mission district of San Francisco (red strokes) and regions that are definitely not part of if (blue strokes), our system automatically determines the visual boundary of the Mission. In addition, we can describe what the visual appearance of the Mission is. For instance, we find that bayview windows, metal grates on doors and windows, dilapidated signs, and bike lanes are all visually discriminative of the Mission. Our predicted boundary agrees more closely with the actual visual boundary of the Mission compared to the "ground truth" boundary.

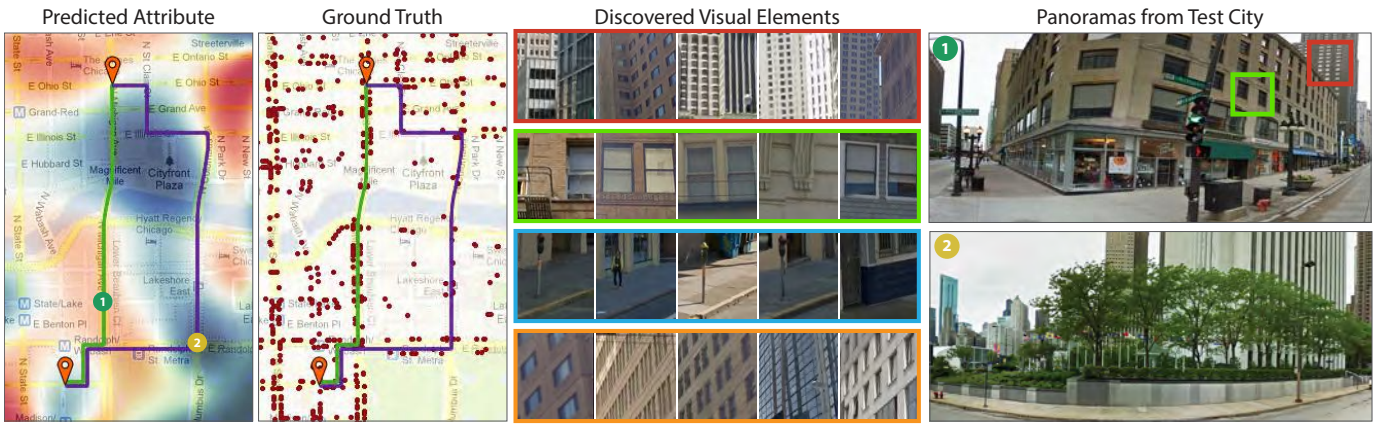


Fig. 11: Compared to a traditional wayfinding route (green path) our theft-avoiding route (purple path) avoids many of the occurrences of theft (red circles) present in the downtown area of Chicago. To compute our route, we predicted the theft rate in Chicago using a predictor trained in San Francisco. The visual elements being detected by the predictor are all associated with high-traffic pedestrian areas, which are likely targets for thieves. In contrast, the route that we compute passes through more open areas with parks and lower foot traffic, which presents fewer opportunities to commit thefts.

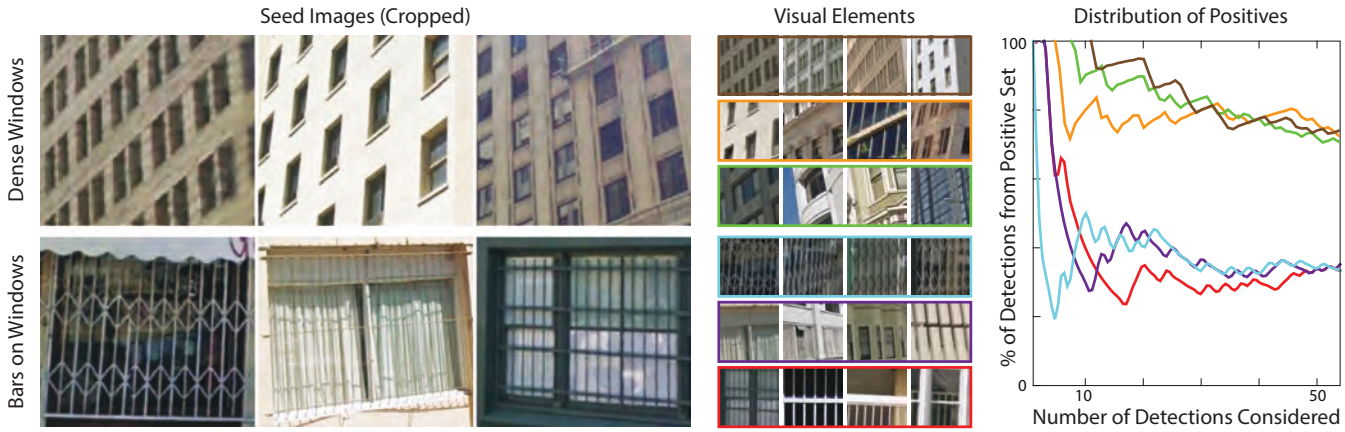


Fig. 12: Our system can be used to validate/invalidate whether bars on windows or high-density windows are predictive of theft. Given the user-provided set of seed images (left) that contain examples of the visual elements, we train a predictor for city attribute "theft rate" in San Francisco using the seed images to initialize the visual element extraction. We then compute the detections of the resulting visual elements for the predictor across the ground truth theft rate values. The percentage of those elements from the positive set to the negative set indicate that bars on windows are significantly less predictive of theft than high-density windows.

[2] A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(1):44–58, 2006.

[3] S. AlHalawani, Y.-L. Yang, P. Wonka, and N. J. Mitra. What makes london work like london. *Computer Graphics Forum (Proceedings of SGP 2014)*, 33, 2014.

[4] J. I. Barredo, M. Kasanko, N. McCormick, and C. Lavalle. Modelling dynamic spatial processes: simulation of urban future scenarios through cellular automata. *Landscape and urban planning*, 64(3):145–160, 2003.

[5] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[6] A. Boyko and T. Funkhouser. Extracting roads from dense point clouds in large scale urban environment. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6):S2–S12, 2011.

[7] M. Carlberg, P. Gao, G. Chen, and A. Zakhor. Classifying urban landscape in aerial lidar using 3d shape analysis. In *Proceedings of the 16th IEEE international conference on Image processing*, ICIP'09, pages 1681–1684, Piscataway, NJ, USA, 2009. IEEE Press.

[8] G. Casella and R. L. Berger. *Statistical inference*, volume 70. Duxbury Press Belmont, CA, second edition, 1990.

[9] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[10] D. Cohen, S. Spear, R. Scribner, P. Kissinger, K. Mason, and J. Wildgen. "broken windows" and the risk of gonorrhea. *American Journal of Public Health*, 90(2):230, 2000.

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

[12] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[13] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes paris look like paris? *ACM Transactions on Graphics (SIGGRAPH)*, 31(4), 2012.

[14] A. Ellaway, S. Macintyre, and X. Bonnefoy. Graffiti, greenery, and obesity in adults: secondary analysis of european cross sectional survey. *BMJ: British Medical Journal*, 331(7517):611, 2005.

[15] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8. IEEE, june 2008.

[16] A. Feuer. The Mayors Geek Squad. New York Times, March 23 2013. `http://www.nytimes.com/2013/03/24/nyregion/mayor-bloombergs-geek-squad.html`.

[17] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[19] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel computing*, 22(6):789–828, 1996.

[20] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[21] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352, 2004.

[22] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

[23] A. Khosla, B. An, J. J. Lim, and A. Torralba. Looking beyond the visible scene. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Ohio, USA, June 2014.

[24] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In J.-O. Eklundh, editor, *Computer Vision ECCV '94*, volume 800 of *Lecture Notes in Computer Science*, pages 189–196. Springer Berlin Heidelberg, 1994.

[25] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

[26] C. A. Latkin and A. D. Curry. Stressful neighborhoods and depression: a prospective study of the impact of neighborhood disorder. *Journal of Health and Social Behavior*, pages 34–44, 2003.

[27] Y. J. Lee, A. A. Efros, and M. Hebert. Style-aware mid-level representation for discovering visual connections in space and time. ICCV, 2013.

[28] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *Image Processing, IEEE Transactions on*, 17(1):53–69, 2008.

[29] V. Mayer-Schönberger and K. Cukier. *Big Data: A Revolution that Will Transform how We Live, Work, and Think*. Eamon Dolan/Houghton Mifflin Harcourt, 2013.

[30] S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler. Tracking groups of people. *Computer Vision and Image Understanding*, 80(1):42 – 56, 2000.

[31] C. A. Micchelli. *Interpolation of scattered data: distance matrices and conditionally positive definite functions*. Springer, 1984.

[32] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, 25(3):614–623, July 2006.

[33] P. Müller, G. Zeng, P. Wonka, and L. Van Gool. Image-based procedural modeling of facades. *ACM Trans. Graph.*, 26(3):85, 2007.

[34] N. Naik, J. Philipoom, R. Raskar, and C. Hidalgo. Streetscore-predicting the perceived safety of one million streetscapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 779–785, 2014.

[35] V. Ordonez and T. L. Berg. Learning high-level judgments of urban perception. In *Proceedings of 13th European Conference on Computer Vision*, 2014.

[36] D. Quercia, N. O'Hare, and H. Cramer. Aesthetic capital: What makes london look beautiful, quiet, and happy? In *The 17th ACM Conference on Computer Supported Cooperative Work and Social Computing*, Baltimore, Maryland, USA, 2014. ACM.

[37] F. Rottensteiner and C. Briese. A new method for building extraction in urban areas from high-resolution lidar data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A):295–301, 2002.

[38] P. Salesses, K. Schechtner, and C. A. Hidalgo. The collaborative image of the city: Mapping the inequality of urban perception. *PLoS ONE*, 8(7):e68400, 07 2013.

[39] J. Secord and A. Zakhor. Tree detection in urban regions using aerial lidar and image data. *Geoscience and Remote Sensing Letters, IEEE*, 4(2):196 –200, april 2007.

[40] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[41] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *Computer Vision–ECCV 2012*, pages 73–86. Springer, 2012.

[42] W. G. Skogan. *Disorder and decline: Crime and the spiral of decay in American neighbourhoods*. University of California Pr, 1990.

[43] A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[44] S. Srinivasan, H. Latchman, J. Shea, T. Wong, and J. McNair. Airborne traffic surveillance systems: video surveillance of highway traffic. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 131–135. ACM, 2004.

[45] P. M. Torrens and A. Nara. Modeling gentrification dynamics: A hybrid approach. *Computers, Environment and Urban Systems*, 31(3):337–361, 2007.

[46] A. M. Townsend. *Smart cities: Big data, civic hackers, and the quest for a new utopia*. WW Norton & Company, 2013.

[47] B. Wahl and E. Wilde. Mapping the world... one neighborhood at a time. *Directions Magazine*, 2008.

[48] J. Q. Wilson and G. L. Kelling. Broken windows. *Atlantic monthly*, 249(3):29–38, 1982.

[49] B. Zhou, L. Liu, A. Oliva, and A. Torralba. Recognizing city identity via attribute analysis of geo-tagged images. In *Proceedings of 13th European Conference on Computer Vision*, 2014.