On Relating Visual Elements to City Statistics

Sean Arietta
Maneesh Agrawala
Ravi Ramamoorthi

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-157
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-157.html

September 10, 2013

Acknowledgement

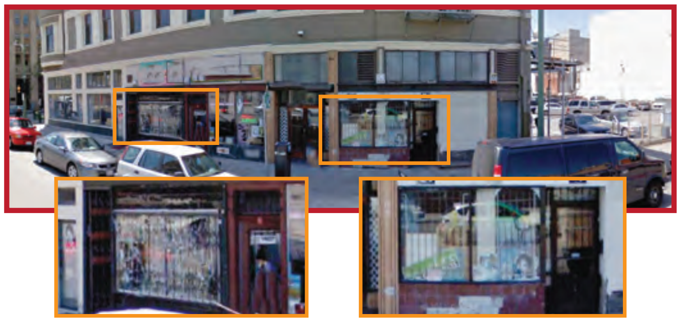
This work was supported by Nokia and the Intel Science and Technology Center for Visual Computing.

On Relating Visual Elements to City Statistics

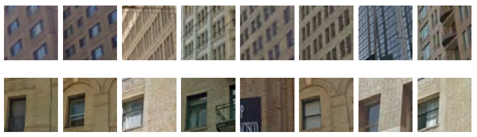
Sean M. Arietta[*]
University of California, Berkeley

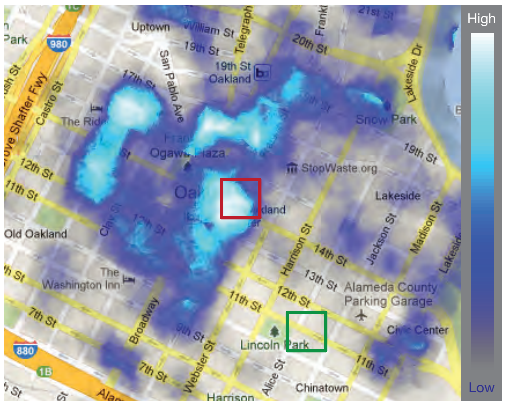
Maneesh Agrawala[†]
University of California, Berkeley

Ravi Ramamoorthi[‡]
University of California, Berkeley


(a) Predicted High Theft Location in Oakland

(b) Predicted Low Theft Location in Oakland


(c) Visual Elements for Thefts in San Francisco



(d) Predicted Theft Rate in Oakland

Figure 1: *Our system automatically computes a predictor from a set of Google StreetView images of areas where a statistic was observed. In this example we use a predictor generated from reports of theft in San Francisco to predict the probability of thefts occurring in Oakland. Our system can predict high theft rate areas (a) and low theft rates area (b) based solely on street-level images from the areas. Visually, the high theft area exhibits a marked quality of disrepair (bars on the windows, unkempt facades, etc), a visual cue that the probability of theft is likely higher. Our method automatically computes machine learning models that detect visual elements similar to these cues (c) from San Francisco. To compute predictions, we use the models to detect the presence of these visual elements in an image and combine all of the detections according to an automatically learned set of weights. Our resulting predictions are* 63% *accurate in this case and can be computed everywhere in Oakland (d) as they only rely on images as input.*

Abstract

We investigate the relationship between visual elements and statistical quantities in cities. Although certain city statistics like the presence of trees and graffiti have a natural connection to visual elements, more abstract statistical quantities such as crime rates and housing prices relate to visual content in a less intuitive way. We show that there is a strong connection between visual elements and these statistics and that this relationship is general enough to predict these statistics in new cities. Given a set of street-level images and geo-located samples of a statistic we first identify visual elements in the images that are discriminative of the statistic (e.g. our system determined that rounded windows and doors in Boston are visually discriminative of affluence). We then build a predictor by learning a weight for each of these elements using a robust regression technique. To perform these operations efficiently, we implemented a scalable distributed processing framework that can process a single statistic (10,000 images) 4x faster than previous methods. We tested the performance of our computed predictors on the statistics: theft, affluence, graffiti presence, and tree presence. We found that at least one predictor for every statistic could interpolate that statistic with 67%-81% accuracy. In addition, we found that we can predict statistics in new cities with up to 76% accuracy. We also tested human performance for predicting theft based on images and found that our method outperformed this baseline with 39% higher accuracy. We present two prototype applications that use our predictors to provide estimates of city statistics: a statistic-sensitive wayfinding program capable of routing travelers through or around statistics of interest (e.g. routing a tourist around a high theft area), and a user-assisted tool for automatically finding graffiti in street-level images.

[*]e-mail:sarietta@berkeley.edu
[†]e-mail:maneesh@berkeley.edu
[‡]e-mail:ravir@berkeley.edu

1 Introduction

How much information about city statistics is present in images alone? If you were asked to decide which of the two locations in Figures 1(a),(b) were most likely to have a high theft rate, which one would you choose? One important difference between these two images is that they contain very different *visual elements*. For instance, Figure 1(a) contains visual elements like fire escapes, storefronts with bars on the windows, buildings in disrepair, etc. In contrast, Figure 1(b) contains visual elements like trees, flowers, etc.

In this work, we investigate the relationship between visual elements and statistical information about cities like theft rates. More specifically, we consider the question of whether there is a measurable relationship between visual elements and the city statistics theft, affluence (measured by housing prices), graffiti presence, and tree presence. We also analyze whether this relationship is general enough to predict statistics in new cities for which samples of these statistics may not be available.

Many applications in city planning, city fund allocation, and wayfinding require a fine-scale representations of statistics like theft, affluence, etc. For instance, if you were to plan a route through a city that avoided high theft areas, you would need (at least mentally) a fine-scale estimate of the likelihood of a theft occurring everywhere in the city. Planning a route sensitive to theft becomes more difficult in areas that you are not familiar with or in areas where theft statistics are not available.

In recent years, one source of information that has become publicly available almost everywhere on the planet and at a very fine scale is street-level imagery (e.g. images in Google StreetView, Bing Streetside, etc). Recently, Doersch et al. [2012] showed that it is possible to automatically identify a set of visual elements that capture the unique visual appearance of cities from these image sources. We extend their approach by automatically computing a set of weights associated with visual elements that are discriminative of a statistic via a robust regression technique called support vector regression [Smola and Schlkopf 2004]. We combine these visual elements and their associated weights into a single "predictor" capable of estimating statistics based solely on images.

To compute and analyze the accuracy of our predictors we process between 30,000 and 170,000 street-level panoramas each of which is approximately 15 megapixels. Processing this many images efficiently presents a non-trivial engineering challenge and one that is often encountered by researchers in the computer graphics and computer vision domains. One of our contributions in this work is the development of a scalable distributed processing framework that can efficiently execute arbitrary computation on a large set of images. In addition to being efficient, we engineered the framework to integrate seamlessly with MATLAB so that researchers could more easily integrate existing systems.

We summarize our main contributions in this work as follows:

1. We describe an extension to the method of Doersch et al. [2012] and Singh et al. [2012] for automatically computing predictors capable of estimating statistics based solely on images (Section 4).

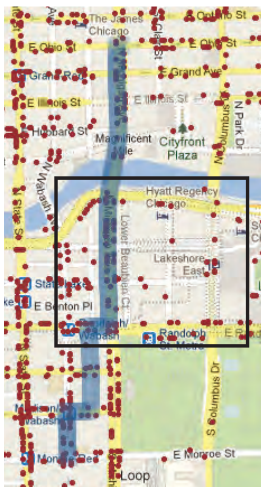
2. We analyze the ability of our predictors to interpolate their respective statistics in cities that were used to compute them. We show that at least one predictor for every statistic can interpolate that statistic with 67%-81% accuracy. In addition, we show that in the case of tree presence in San Francisco, for which we could actually compute ground truth via Mechanical Turk, we are able to achieve 10% higher accuracy than a radial basis function interpolation (Section 6.1).

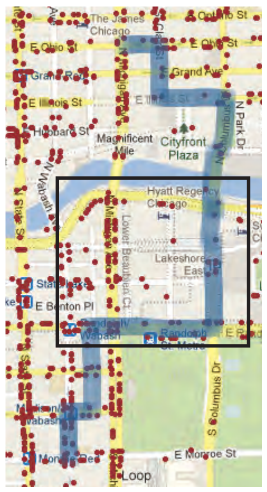
3. We analyze the ability of these predictors to generalize to new cities. We summarize these results in Figure 13. Of particular note is our ability to predict the theft rate in downtown Chicago from visual elements of theft in San Francisco with 39% higher accuracy than humans participating in a Mechanical Turk study (Section 6.2).

4. We present two prototype applications that rely on our predictors to compute a fine-scale representation of certain statistics (Section 7). In Figure 2 we show an example output of a wayfinding application we built that allows a user to traverse a path through or around a statistic of interest - in this case around theft. Other statistics can be used in the system as well, for instance trees (Figure 14). We also built a system that allows a user to quickly locate areas where graffiti is likely to occur in a city using predictors trained in that city. A result using this system is presented in Figure 15.

2 Related Work

The connection between visual elements and city statistics has been considered in a variety of different fields. One of the earliest and most well-known examples of a principled study of this connection is the the famous "broken window theory" [Zimbardo 1969; Wilson and Kelling 1982]. The theory states that if a car or building with broken windows is left unattended, the remaining windows are much more likely to be broken. In essence, at least some of the visual elements of cities (e.g. cars and buildings) contribute significantly to the rate of crime that is observed. This connection between visual elements and city statistics has been shown to hold not just for crime, but for a variety of other statistics including obesity [Ellaway et al. 2005], depression [Latkin and Curry 2003], and sexually transmitted diseases [Cohen et al. 2000]. Researchers in more recent years have even used Google StreetView to conduct visual "audits" of statistics in areas of interest either by their own inspection [Rundle et al. 2011] or via crowd-sourcing [Hara et al. 2013].


(a) A Typical Route


(b) A Theft-Avoiding Route

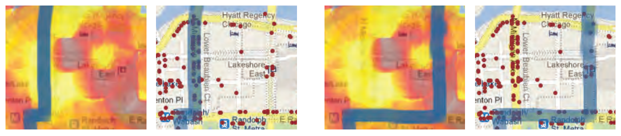


Figure 2: *Figure 2(a) shows a route generated from a typical wayfinding implementation that attempts to minimize the total travel distance from a source to a destination. In Figure 2(b), the user has indicated that he/she wishes to avoid avoid high theft-rate areas in Chicago between the same source and destination. The left insets show the two routes in more detail, overlayed on our estimates for the probability of theft (light red is low, bright yellow/white is high) based on a predictor trained in San Francisco. Although we are using a predictor from a different city, the route still avoids the thefts that actually did occur in this area (red circles).*

Although researchers have shown that these relationships exist, they have had to rely on their own manual inspection of image data to derive them. In contrast, our method relies only on samples of statistical data and an associated set of street-level images to automatically learn the set of visual elements that are discriminative of that statistic. In addition, we are able to combine detections of these visual elements allowing us to predict statistics in new areas at the density of street-level images (usually around 3-4m along streets in cities), something that would be intractable from a set of human observations alone.

Researchers have also analyzed the relationship between visual elements and the unique visual appearance of cities [Doersch et al. 2012]. For instance, Paris has a very distinctive visual appearance compared to London. Our approach leverages the same clustering approach as their work (developed by Singh et al. [2012]) to extract machine learning models that detect discriminative visual elements. We extend their technique by computing a set of weights over the

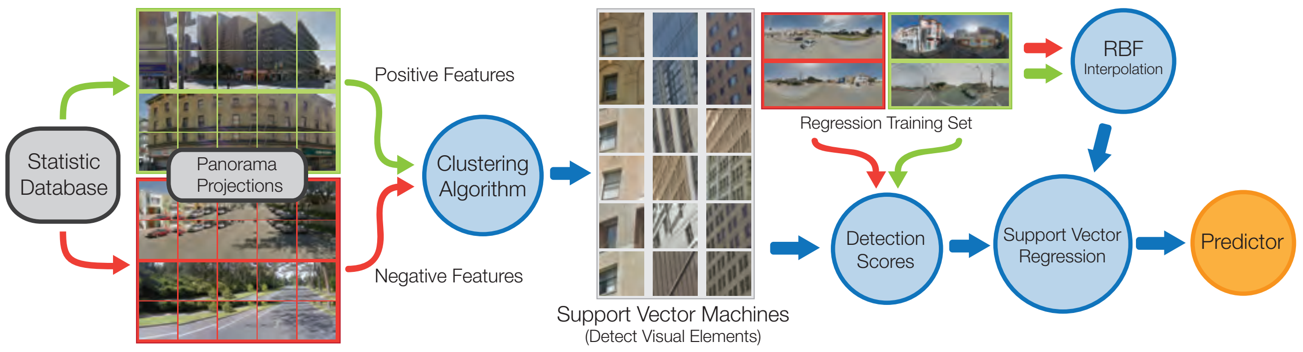


Figure 3: *To compute a predictor, we first download a set of of statistics from a "Statistic Database" that have been annotated with latitude and longitude coordinates. We then compute a set of positive and negative image features from Google StreetView panoramas at the locations of the statistics. We use the image features in a "Clustering Algorithm" designed to recover visual elements that are discriminative of the statistic. One of the outputs of this algorithm is a set of "Support Vector Machines" (SVMs), each of which can detect the presence of a visual element in an image. We use these SVMs to detect the presence of the visual elements in a set of images distinct from those used in the clustering algorithm. We also interpolate the value of the statistic at the locations of these images. We use "Support Vector Regression" to find weights over the visual elements such that their weighted detections match the interpolated values. The set of SVMs and their associated weights are what we refer to as a "Predictor".*

models to produce a predictor. This extra step is necessary for two reasons. First, the resulting models from the clustering approach only provide a binary detection of visual elements (e.g. does this image contain a window with bars on it or not). Second, there are potentially hundreds of possible models that are produced via the clustering technique. Knowing which subset of those is actually indicative of a particular statistic and how to combine them in a principled way has not been considered in previous work.

We solve both issues using a regression technique that computes the weight of each model while removing models that are redundant or unneeded by setting their weight to 0. The specific regression technique we use is called support vector regression. This technique is robust to overfitting and explicitly attempts to compute weights that remove unnecessary dimensions (i.e. redundant and unneeded detections) [Boser et al. 1992; Smola and Schlkopf 2004]. Support vector regression has been applied in a number of different fields including graphics for 3D pose estimation [Agarwal and Triggs 2006] and image restoration [Mairal et al. 2008].

There have been several approaches that use imaging modalities other than street-level panoramas to predict and detect city statistics. Aerial and street-level lidar has been used to detect trees [Secord and Zakhor 2007], water and ground coverage [Carlberg et al. 2009], roads [Boyko and Funkhouser 2011], and buildings [Rottensteiner and Briese 2002]. Video has been used to categorize traffic in cities [Koller et al. 1994; Srinivasan et al. 2004] as well as to track crowds for the purposes of detecting flow patterns, anomalous behavior, etc [McKenna et al. 2000; Hu et al. 2004]. Although these techniques often produce very accurate results, the availability of these sources of data is a limiting factor in most cities. Since our method relies only on street-level images, we have almost no limitation on the places we can use our predictors.

3 Overview

Our goal is to create a predictor that can estimate the probability of observing a statistic in an area based on street-level images from the area. We define a predictor as a set of support vector machines (described in Section 4) that detect visual elements and a set of weights associated with detections. Figure 3 shows an overview of our approach and includes five steps:

1. We download occurrences of a city statistic that have been annotated with latitude and longitude coordinates from an online "Statistic Database" (Section 3.1).
2. We compute a set of image features from perspective projections of Google StreetView panoramas where the statistic occurred (positive features) and where the statistic is unlikely to have occurred (negative features) (Section 3.2).
3. We train a set of support vector machines (SVMs) that detect discriminative visual elements using a "Clustering Algorithm" on the positive and negative features (Section 4.1).
4. We collect a "Regression Training Set" of panoramas not used to train the SVMs and interpolate them via radial basis functions.
5. We compute a set of weights over the detections of the SVMs using "Support Vector Regression" to match the interpolated values (Section 4.2).

3.1 Acquiring the Statistics

In our experiments we downloaded statistics that were annotated with latitude/longitude coordinates for (1) occurrences of theft, (2) affluence as measured by the top 20% most expensive homes in dollars per square feet, (3) the presence of graffiti, and (4) the presence of trees. Note that each of these is a binary value indicating where the the statistic occurred. We downloaded statistics (where available) in: San Francisco, Oakland, Seattle, Chicago, Los Angeles (Inglewood neighborhood), Boston, and Philadelphia.

For crime data, we used two resources freely available on the Internet: CrimeMapping.com and CrimeReports.com. From these sites we were able to acquire the occurrences of crimes for the past year in all cities. The housing price data came from the online real estate company Zillow. We were able to find this information for all of the cities we investigated except in the Inglewood neighborhood (Los Angeles). We limited our data to only homes that have sold in the past few years rather than relying on for-sale prices which don't necessarily represent the value of a home. We found reported graffiti incidents in San Francisco[1], Boston[2], and Chicago[3]. We were only able to find one data source for tree presence in one city - San Francisco (UrbanForestMap.org).

3.2 Image Features

Both the clustering algorithm and our predictors operate on image features rather than entire street-level images. We extract these image features from perspective projections of Google StreetView panoramas. We project each panorama at 20° intervals across the

[1]https://data.sfgov.org
[2]https://data.cityofboston.gov
[3]https://data.cityofchicago.org

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 (Figure 3).

We compute features using the same method as Doersch et al. [2012]. We first scale each panorama projection to a resolution of 400x400 pixels. We construct a Gaussian pyramid from each downsampled image with an 8x8 patch at the coarsest level. For each level in the pyramid we compute HOG features [Dalal and Triggs 2005] over the entire level. We concatenate these features with the (a^*, b^*) coordinates (in the CIE La*b* color space) of the original image downsampled to the resolution of the pyramid level. An image feature is an 8x8 patch of the concatenated features from one of pyramid levels. This produces features of dimension 2112.

To compute our predictor we split the image features into a positive set and a negative set. If the image feature is from a panorama that is co-located with an occurrence of the statistic (e.g. theft, affluence, etc), we put the feature in the positive set. Defining the negative set of features requires more work since more of the data on city statistics only specifies where the statistic occurred.

To generate the negative set our goal is to extract features from panoramas that correspond to locations (x,y) where a statistic is *unlikely* to be observed. Since the set of all locations that meet this criterion could produce more locations than our system can handle, we define a probability distribution over the negative locations and generate the necessary number of samples from it.

To generate the samples, we first compute a 2-dimensional histogram over the positive locations. Since each statistic is a binary value, every positive location (i.e. a latitude/longitude coordinate of an occurrence of the statistic) contributes a value of 1 to the histogram. We use a bin size of 0.0005x0.0005 with respect to latitude and longitude. We then blur this histogram with a Gaussian filter ($\sigma = 3$) to build a continuous probability distribution $g(x,y)$.

We define the probability that a location (x,y) should be considered a negative as:

$$P[(x,y) \text{ is negative location}] = \begin{cases} 1 & \text{if } g(x,y) < t \\ 0 & \text{otherwise} \end{cases}$$

We set the threshold t to 0.0001 in all our experiments. To generate negative locations, we sample this probability distribution using the common inverse cumulative distribution sampling method, more formally referred to as the probability integral transformation [Casella and Berger 1990]. Panoramas from these locations generate the negative set of features. An example of the positive and negative locations used for the theft statistic in San Francisco is shown in Figure 4.

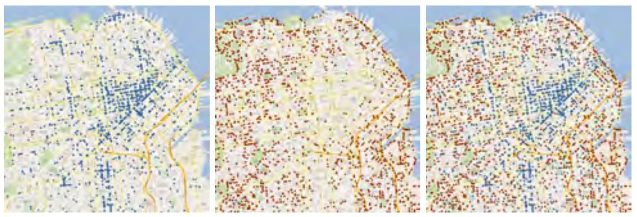
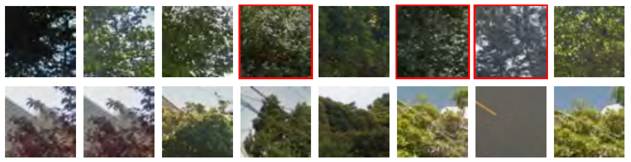

(a) Positive Samples (b) Negative Samples (c) Overlayed Samples

Figure 4: *An example of locations generated for the theft statistic in San Francisco. Negative locations, which are often not measured, are defined as locations where the statistic is unlikely to occur. Panoramas from these locations supply the negative image features.*


(a) Two examples of higher-ranked nearest neighbor sets for the tree presence experiments in San Francisco.

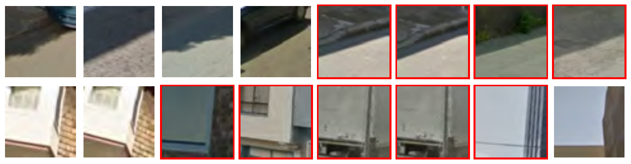

(b) Two examples of lower-ranked nearest neighbor sets for the tree presence experiments in San Francisco.

Figure 5: *A few examples of the nearest neighbor sets used to seed the clustering process for the tree presence statistic in San Francisco. The first two rows are higher ranked based on the number of nearest neighbors that are from the positive set versus the negative set. The bottom rows are some of the lowest rankings. Although the nearest neighbor does find good examples of the statistics, it lacks enough uniformity to be used directly to detect a single visual element.*

4 Building the Predictor

To analyze the relationship between visual elements and city statistics we compute a predictor that is capable of estimating locations where a statistic is likely to occur based on street-level imagery. The predictors that we compute are comprised of (1) a set of support vector machines (SVMs), each of which detects the presence of a single visual element discriminative of a statistic, and (2) a set of weights associated with the SVM detections.

An example of the visual element "high density apartment windows" in Chicago is shown in the top row of Figure 6. Each row represents a visual element and each image in the row represents a single detected region in a panorama from Chicago. These detections were computed using an SVM trained using our system. Since all of the detections are examples of high density apartment windows, we say that the SVM detects the presence of this visual element.

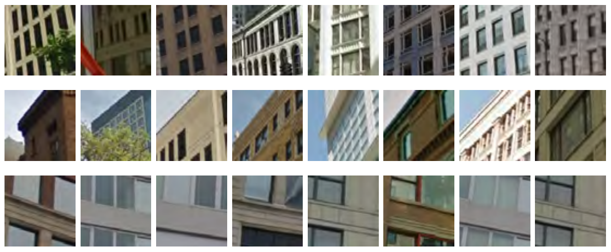
4.1 Computing the Visual Element Detectors

In this work we use linear SVMs which attempt to find a hyperplane (defined by a vector normal to the plane $\mathbf{n}$ and an offset from the origin o) that separates a particular set of features from another set of features with the largest separation distance. We refer to the computation of these hyperplanes as *training* the SVM. Our goal is to train SVMs that separate features corresponding to a single visual element from all the other features and that detect visual elements that are discriminative of a particular statistic.

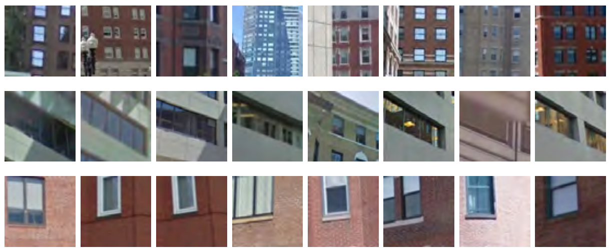
Singh et al. [2012] have recently developed a method for computing linear SVMs that satisfy these two conditions. Their technique uses an iterative approach to find clusters of image features that represent a single visual element and an associated linear SVM capable of detecting whether that visual element is present in an image. We use a version of this clustering technique implemented by Doersch et al. [2012] to train our linear SVMs.

To initialize the clustering algorithm, we randomly sample 25,000 positive image features (defined in Section 3.2) and compute their nearest neighbors in feature space across the entire dataset (positives and negatives). Each set of nearest neighbors becomes an initial cluster.

To reduce the number of clusters that we have to process, we sort



(a) Visual Elements for Theft in Chicago



(b) Visual Elements for Theft in Boston

Figure 6: *Some of the visual elements for the theft statistic in Chicago and Boston. In both cities, high-density windows and windows with no extra additions (balconies, planters, window sills, etc) are likely to be detected where thefts occur. Although these visual elements may not seem intuitively indicative of theft, the resulting predictors are both capable of estimating the probability of crime with nearly 80% accuracy.*

the initial 25,000 positive features based on the number of their closest 20 nearest neighbors that came from the positive feature set and only use the top 800 nearest neighbor sets as the initial clusters. This ranking helps to produce good initial candidate clusters because a set of nearest neighbors that consists mostly of positives is likely to be discriminative of the statistic. If a set of nearest neighbors is not discriminative they will be more randomly distributed amongst the positive and negative set.

Figure 5 shows some of the nearest neighbor sets that we use to define the initial clusters in our tree presence experiments (Section 6). The nearest neighbor sets in Figure 5(a) are ranked higher than the nearest neighbor sets in Figure 5(b). Red boxes indicate nearest neighbors from the negative set. The higher ranked clusters are more visually similar to the kinds of visual elements that are discriminative of trees.

The details of the iterative clustering algorithm that we use are the same as Doersch et al [2012] and Singh et al. [2012]. We elide the details of this method for brevity, but summarize the critical steps in Figure 7. We represent each cluster as an SVM that detects whether an image feature is a member of the cluster. To refine the clusters we iteratively compute the "SVM Detections" across the negative set and then "Re-Train" the SVMs so that they avoid detecting these negatives. This approach is referred to as the hard-negative mining technique [Felzenszwalb et al. 2008] and helps to quickly refine the SVMs so that they don't erroneously label negative features as positive features. The SVMs associated with the final set of clusters are what we use to construct our predictors.

Figure 8 shows some examples of the initial and final iterations of this approach. Note that the initial clusters from just the nearest neighbor sets contain many members from the negative set and lack a consistent appearance. We run the iterative approach for 2 iterations, after which we find the final clusters exhibit a lot more uniformity and more members from the positive set.

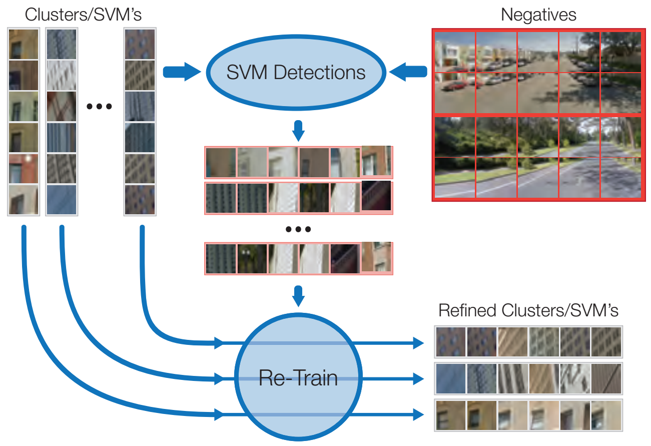


Figure 7: *A detailed view of the iterative nature of the clustering algorithm. We represent each cluster as an SVM capable of determining whether an image feature is a member of the cluster. The SVMs are used to compute detections in the negative set. The resulting detections are used to re-train the clusters to avoid detecting negatives that the SVM is likely to erroneously detect as positives. The refined SVMs are then fed back into this loop for further refinement.*

4.2 Computing the Predictor

Statistics are represented by a complicated combination of many visual elements, not the presence (or lack thereof) of a single visual element. Thus, we cannot directly use the linear support vector machines (SVMs) trained by the clustering algorithm to predict them. Much in the same way that attribute classifiers [Kumar et al. 2009] are built from a set of simpler classifiers, we compute our predictors by learning a set of weights over the SVM's *detection scores*. By combining the weighted detection scores of each SVM, we can compute an estimate of the probability of observing a statistic based on an image's content.

Since the SVMs are linear, we can compute the detections scores as:

$$d_{i,j} = \mathbf{f}_i \cdot \mathbf{n}_j - o_j \tag{1}$$

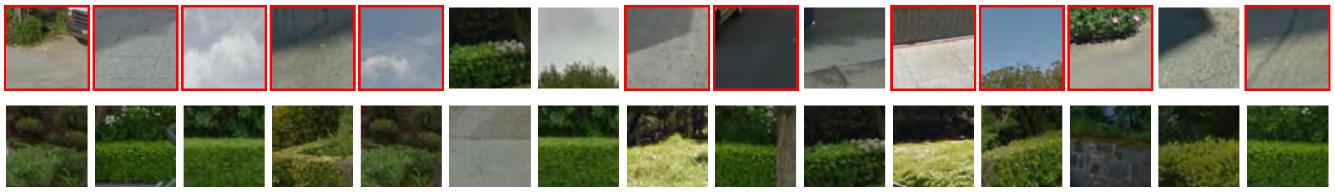
Where $d_{i,j}$ is the score for SVM $(\mathbf{n}_j, o_j)$ on image feature $\mathbf{f}_i$.

The particular method we use to determine the weights is called support vector regression [Smola and Schlkopf 2004] and is intentionally engineered to be sensitive to over-fitting. This is an important issue in our context since the raw detection scores are not meant to represent the probability of observing a visual element and thus can be unreliable.

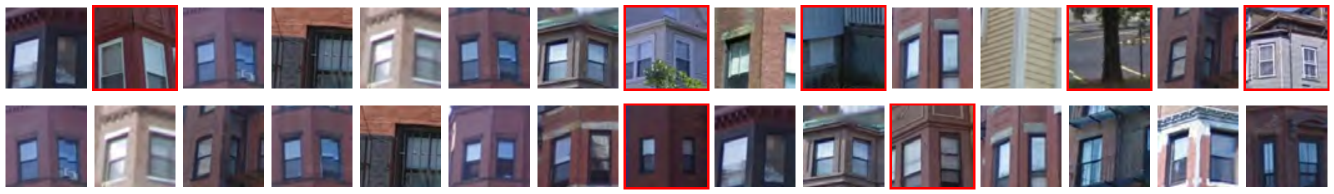
In SVR, the goal is to find a function that approximates a set of training outputs y_i given a set of training input vectors $\mathbf{x}_i$. In our case, each coordinate of a training input vector is the detection score of a single SVM in a single image of the "Regression Training Set" depicted in Figure 3. In our experiments we use the top 3 detection scores from each SVM to keep the optimization well conditioned.

To generate the training outputs y_i, we interpolate the original statistic locations at the locations of the panoramas in the "Regression Training Set" using radial basis functions (RBFs). To compute the basis functions, we initialize a multiquadric RBF with weight 1 at each of the original statistic locations. These RBFs have the following form:

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



(a) **Top:** An initial cluster for affluence (houses worth more than \$850/sqft) in San Francisco. **Bottom:** The corresponding final cluster.



(b) **Top:** An initial cluster for affluence in Boston (houses worth more than \$500/sqft). **Bottom:** The corresponding final cluster.

Figure 8: *A visualization of the iterative clustering technique. The goal of the clustering technique is to compute a set of linear support vector machines that can detect visual elements that are likely to be present in images containing a house worth more than \$850/sqft in San Francisco and \$500/sqft in Boston. The top row in each example is the set of patches comprising the initial cluster (the nearest neighbor set). The bottom rows are the final clusters after the iterative approach. Image features that are part of the negative set are highlighted in red. The clustering approach removes negative examples and produces clusters (and their associated SVMs) with more consistent detections that are visually related to the statistic (e.g. shrubs in Francisco and protruding window facades in Boston).*

We tried several different RBFs but found that the multiquadric RBF with $\epsilon = 2$ worked well in practice.

We can parametrize the functions that approximate the y_i's given the $\mathbf{x}_i$'s as $f(\mathbf{x}_i) = \mathbf{w} \cdot \phi(\mathbf{x}_i) + b$, where $\phi(\mathbf{x}_i)$ is a (possibly non-linear) map on the $\mathbf{x}_i$'s that can help find a better approximation. The $\mathbf{w}$ in this parametrization is what we refer to as the weights in the predictor. In the simplest case, ϕ is just the identity map and $f(\mathbf{x}_i)$ reduces to the standard equation for a hyperplane. Figure 9 shows a visualization of these quantities and the relevant parameters of the approach in a simplified 2-dimensional case.

The goal of SVR is to determine the parameters $(\mathbf{w}, b)$ that are as "compact" as possible while minimizing the loss of a particular parametrization $(\mathbf{w}, b)$ in predicting the training outputs y_i. This loss is defined as:

$$L_\epsilon(y_i, (\mathbf{x}_i, b)) = \max(|y_i - f(\mathbf{x}_i)| - \epsilon, 0) \tag{3}$$

Here, we have introduced a new parameter ϵ which controls the magnitude of error that we tolerate in our fitted model. Any y_i that are within a distance ϵ of the fitted model are considered equally well approximated. In Figure 9 this property means that no y_i in the gray shaded region contribute to the error of the fit. We set ϵ to be 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 $\mathbf{w}$ subject to the constraints imposed by the loss function.

One important detail here is in the selection of the map ϕ, which typically transforms input points into a higher dimensional space enabling non-linear regression. In order to define this map ϕ, we need only to define a so-called "kernel function" $K(\mathbf{x}_i, \mathbf{x}_j)$ operating on pairs of training inputs. We considered the following three forms of K:

$$\begin{aligned}
\text{Linear:} \quad & K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \\
\text{Polynomial:} \quad & K(\mathbf{x}_i, \mathbf{x}_j) = \left(\gamma \mathbf{x}_i^T \mathbf{x}_j + r\right)^d \\
\text{Radial Basis Function:} \quad & K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2\right)
\end{aligned}$$

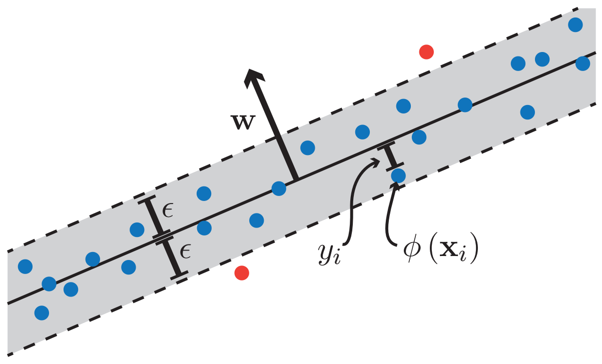


Figure 9: *A visualization of the support vector machine regression optimization in 2D with the linear map $\phi(\mathbf{x}_i) = \mathbf{x}_i$. The goal of the optimization is to find a hyperplane defined by $(\mathbf{w}, b)$, such that all of the distances from this plane to the training data points $\mathbf{x}_i$ are within ϵ of their actual distance given by y_i. The red points in the diagram illustrate points that don't fit this criterion and therefore contribute to the error or loss of the fit.*

All of our experiments use the radial basis function form with γ equal to $1/D$, where D is the number of dimensions in the training vectors x_i. It's worth noting that the radial basis function used in regression is unrelated to other references in the paper regarding radial basis functions used for interpolation, although they all have this basic form.

The resulting $(\mathbf{w}, b)$ along with the associated SVMs define a predictor. We use the popular library `libsvm` [Chang and Lin 2011] to perform the support vector regression step and to compute predictions in new images. We use the default settings of the library unless otherwise noted. It's important to note here that this predictor can be used in *any image*, which means we can predict the occurrence of statistics anywhere there is street-level imagery available regardless of the city that was used to generate it (results and applications are presented in Sections 6 and 7 respectively).

5 Implementation

Computing the linear support vector machines (SVM) via the clustering method described in Section 4 requires performing a number of compute and data-intensive operations. The input to the algorithm in our experiments is a set of 10,000 Google StreetView projections (2,000 positive locations, 8,000 negatives), each of which is 640x640. Every image contributes 7,700 image features, each of which is comprised of 2112 floating-point numbers. This amounts to about 650GB worth of image data that we must process. In addition, *each iteration* of the algorithm depicted in Figure 7 can require upwards of 26GB to be processed and around 2GB to be transferred between the detection step and the re-training step. When we compute predictions of our statistics in new cities we have to compute detections for all of the models that comprise the predictor in every street-level image in the city, which ranges from 30,000 to 170,000 panoramas that have a resolution of 5324x2867 each. This is nearly 2.5TB worth of data that must be processed in some cases.

In order to efficiently process all of this data, we developed a scalable distributed visual processing framework. We have used our framework on both a local heterogeneous 40 core cluster and on Amazon's Elastic Compute Cloud. Our framework can compute the SVMs via the clustering method described in Section 4.1 for a single statistic in about 12 hours on a cluster with 40 total cores (480 CPU hours). This is a 3.75x speedup over the previous implementation [Doersch et al. 2012]. We can compute predictions over an entire city of 80,000 panoramas in 196 CPU hours, or about 5 hours on our 40 core cluster.

Although there have been a number of implementations for performing distributed tasks on a heterogeneous/homogeneous cluster of computers, our implementation leverages certain assumptions about common visual processing tasks to increase efficiency that more general frameworks cannot. Specifically, we assume that all data being passed between processors can be encoded in a matrix and that the order of that data is inconsequential. This last assumption means that the matrices don't need to be sorted, shuffled, or reduced, which is a common costly operation in many popular processing frameworks [Dean and Ghemawat 2008; White 2012].

We implemented our framework in C++ and used the standard message passing interface (MPI) library as our communication protocol. In order to make our framework easier to use in the scientific community, we added support for MATLAB matrices to be used directly in our system. In fact, all variables passed between nodes are wrapped in a MATLAB matrix, with extra functions to make serialization and deserialization efficient.

We added several features to our framework to make it more robust and efficient. We briefly outline these below.

Scheduling: The main purpose of our framework is to maximize the throughput of tasks. Since not all pieces of a task necessarily take the same amount of time and not all nodes are guaranteed to process tasks at the same rate, scheduling becomes an important aspect of distributed processing. Our restriction that all variables must be expressed as matrices allows us to easily break up inputs by columns or rows and iteratively send these out to be processed. Nodes that are slow will simply receive fewer rows/columns. This avoids a very common issue in first-order implementations of distributed processing frameworks wherein a job's execution time is equal to the execution time of the slowest node.

Checkpointing: Whenever enough of an output matrix has been computed, the partially completed matrix can be saved to disk. If a job dies during execution for any reason, it can be restarted where it left off by loading the checkpoint before resuming computation.

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 filesystem. 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 metadata), the framework can instruct the nodes to load the variables from the cache rather than re-transferring them.

Partial Variables: In many visual processing tasks a large output is computed. For instance, the result of computing the detections from the negative set (the "Cluster Detections" block of Figure 7) can easily require several dozen gigabytes to be in use on the master node of the cluster. To avoid using this much memory, a variable can be marked as "partial" indicating that either rows or columns of the associated matrix can be saved to disk once they have been computed.

6 Results

We use the predictors generated via the method described in Section 4 to analyze the relationship between visual elements and city statistics. Recall that our predictors are comprised of a set of linear support vector machines (SVMs) that detect visual elements that are discriminative of a particular statistic along with a set of weights that encode the importance of each of the detections.

Our results indicate that in many cases there is a quantitatively relevant relationship between visual elements and city statistics. In Section 6.1 we analyze the strength of this relationship by interpolating theft, affluence, graffiti presence, and tree presence in cities using predictors generated in the same city. To evaluate the ability of these relationships to generalize to new cities, we analyze the performance of predicting statistics in "target" cities based on predictors trained in "source" cities. Results are presented in Section 6.2. We used San Francisco, Chicago, and Boston as source cities and San Francisco, Oakland, Seattle, Chicago, Los Angeles (Inglewood neighborhood), Boston, and Philadelphia as target cities, although some results don't contain all target cities because the data was not always available.

The four statistical quantities that we tested were:

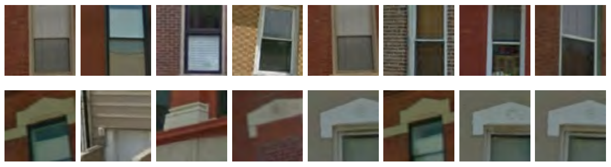
1. Theft rate measured as locations where thefts have occurred in the past year.
2. Affluence measured as locations where homes have sold for more than 80% of all sold homes in a city in the past several years.
3. Graffiti presence measured as locations where graffiti has been reported as being offensive.
4. Tree presence measured as locations where people have noted a tree that was larger than 70 centimeters ($\sim$ 27 inches).

Some examples of visual elements that we automatically determined were discriminative of theft are shown in Figure 6, affluence in Figure 8, and graffiti in Figure 10. More resulting visual elements are available in the supplementary material.

The evaluation metric we use in our analysis is the area under the receiver operator characteristic (ROC) curve. The ROC curve is defined as the relationship between the true positive rate (TPR) and the false positive rate (FPR) of a binary classifier for any prior distribution on the negatives and positives. The true positive rate and the false positive rate are related to the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) of a binary classifier compared to ground truth via the following two equations:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \qquad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} \tag{4}$$

To determine true positives, false positives, etc. we derived ground truth estimates for each of the statistics in every city the same way we generated positive and negatives samples used during training (see Section 3.2 for details).


(a) Top Visual Elements for Graffiti in Chicago

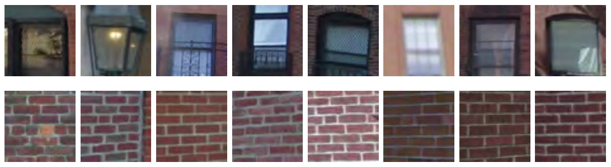

(b) Top Visual Elements for Graffiti in Boston

Figure 10: *(a) shows some of the top visual elements for the graffiti statistic in Chicago. The second row in (a) is actually an architectural style that is very common around the "East Village" neighborhood of Chicago, which we only realized upon inspecting the source locations of these images. (b) shows some top elements from Boston. The bottom row appears to be just bricks, however, when we inspected their corresponding locations we found nearly all of them were in alleyways.*

This metric is useful in our context as it measures the probability that a random positive sample is classified correctly with a higher confidence than a randomly chosen negative sample. Moreover, it has been shown that if the area under the ROC curve is equal to 0.5 then the classifier is not discriminative at all i.e. it's a random classifier, regardless of the prior distribution on positives and negatives. Anything above 0.5 is considered discriminative with some accuracy up to a maximum value of 1.0 [Hanley and McNeil 1982].

We convert our predictions to binary classifications by simply classifying predictions above a certain value as positive and those below as negative. To produce the curves in our plots, we varied this threshold from 0 to 1 and computed the true positive rate versus and false positive rate for each threshold value.

The results of our experiments are summarized in Tables 1 (graffiti presence) , 2 (theft), and 3 (affluence), and in Figures 11 (tree presence) and 13 (ROC curves). Each entry in the tables represents the area under the ROC curve of a source/target city pair. Diagonals of the tables correspond to the intracity interpolation experiments and represent the area under the bolded ROC curves in Figure 13.

6.1 Intracity Prediction Results

In our first set of experiments, we analyze the ability of predictors generated in a source city to predict where statistics are likely to be observed in that same city, but at different locations than were used to compute the predictors. We are effectively testing the ability for our predictors to interpolate statistics based on images alone. We considered the four statistics theft, affluence, graffiti presence, and tree presence in San Francisco, Boston and Chicago.

In almost all cases the area under the ROC curves for every city and statistic pair was above 68% prediction accuracy with several around 77%. The best interpolation result was the affluence statistic in Boston (Table 3), which was 81.5% accurate. To provide an intuition for the significance of these numbers, we deployed a Mechanical Turk experiment asking workers to examine a series of 15 Google StreetView images from Chicago 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 970 images. For each image we averaged the responses of 5 workers and thresholded the averages to generate the ROC curves in Figure 13 in the row for theft and the column for Chicago. Our predictor's interpolation of this statistic outperforms the human performance by 39%.

The only statistic/city pair that did not perform well was graffiti in Chicago. We believe that this is most likely due to the fact that we are recovering visual elements that are correlated to graffiti rather than representative of it (see Figure 10). Notice that none of the detections for the visual elements contains graffiti. Rather the visual elements seem to be attributes of the cities that are likely to be present in areas where graffiti would be observed (e.g. alleyways, particular neighborhoods). According to our analysis, this correlation is seemingly weaker in Chicago than in San Francisco and Boston.

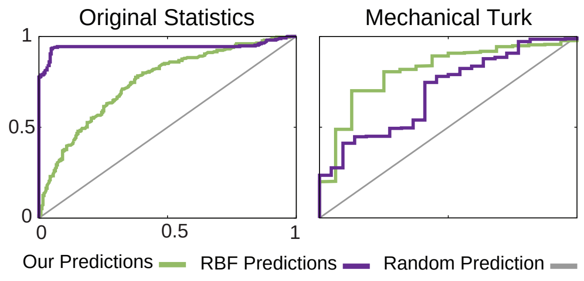
	Target Cities		
Source Cities	San Francisco	Chicago	Boston
San Francisco	0.682	0.553	0.615
Chicago	0.492	0.559	0.550
Boston	0.560	0.496	0.686

Table 1: ***Graffiti Prediction Performance:*** *Graffiti relates to visual elements with lower performance than our other statistics. While graffiti has an obvious visual appearance, our algorithm is not able to capture the necessary fine-scale details. With better image features or more specific training examples, the performance of our method would likely improve.*

6.1.1 Errors in the Ground Truth Data

One of our motivations for pursuing this research is that online databases of city statistics can be very unreliable. Figure 11 shows both the ROC curves and the area under those ROC curves for the tree presence statistic. The green curve in leftmost plot of Figure 11 represents our method with respect to ground truth derived via the method described in the introduction to this section. Compared to a radial basis function interpolation of the statistic (purple curve), our method seemingly does much worse. However, when we asked Mechanical Turk workers and Facebook friends to decide whether a set of images contained "at least 50% of a single tree" and compared our method versus radial basis functions to their answers, we found that our predictions were more than 10% more accurate than the radial basis function interpolation. In this experiment we tested 1000 images and an average of 10 people marked each image.

This result is an important illustration of the shortcomings of online



	Original Statistics	Mechanical Turk
Our Predictor	0.750	0.811
RBF Predictor	0.944	0.698

Figure 11: *For the second column of the table (corresponding to the ROC curves in the left plot) the ground truth was derived from the statistical information downloaded from UrbanForestMap.org. In this case a radial basis function interpolation of the statistic performs exceptionally well. However, when we generated a dataset from a crowd-sourced labeling experiment (right plot and third column of the table), our predictor showed a 10% performance gain over the standard interpolation approach.*

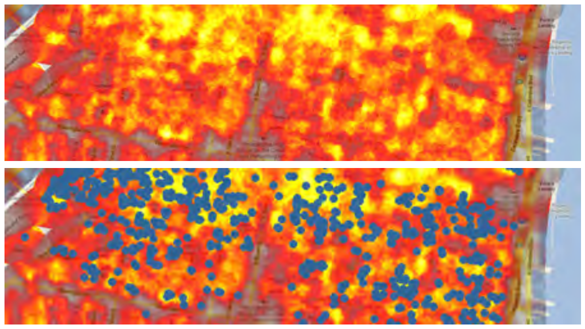


Figure 12: *The top image shows a heat map of our predictions of affluence in Philadelphia based on a predictor trained in Boston. On the bottom we have overlayed the actual locations of affluence (measured by the price of homes). The predictions are very similar to the ground truth data despite the fact that the predictions rely solely on images and were trained in a different city.*

databases of statistics; sometimes they are imprecise or difficult to interpolate in a principled way. Fortunately, our predictors rely only on images and thus can potentially outperform these standard interpolation approaches.

6.2 Cross-City Prediction Results

For the statistics theft, affluence, and graffiti presence, we estimated the probability that the statistic would be observed in a target city based on a predictor trained in a source city (i.e. cross-city predictions).

For every statistic, we found at least one example of a predictor that generalized well to a new city. For the theft statistic (Table 2), the predictor trained in Chicago was 76% accurate in San Francisco, almost as accurate as the predictor trained in San Francisco. Figures 1 and 2 shows two qualitative examples of our theft predictions.

In Table 3 we summarize the performance of our cross-city predictions for affluence. The affluence predictor for Boston shows a strong generalization to Philadelphia, which seems reasonable considering both cities were founded at roughly the same time (mid 15^{th} century) and exist in the same general geolocale. Figure 12 shows these predictions overlayed with the actual locations of affluence in Philadelphia. Our predictions are very accurate in this case despite being estimated from images.

Graffiti had the lowest cross-city prediction performance. We attribute this to two factors. First, the visual elements that we determined are discriminative of graffiti are all correlated to graffiti rather than actually representative of it (see Figure 10). Second, graffiti occurs at very specific locations in an image, but the statistics we use only specify the latitude and longitude where graffiti was reported. Thus, we can't provide very specific examples of what graffiti looks like, only what the areas that contain graffiti look like. If we had access to more precise locations our predictors would likely generalize better.

The Los Angeles crime statistic was notably difficult to predict. When we looked into the nature of thefts in the area of Los Angeles we considered, we noticed that the ground truth theft data was always sampled at intersections instead of at specific latitude/longitude coordinates, which may have contributed to our poor performance. This lack of fine-scale statistical samples is one of the motivations for this work.

The Mechanical Turk study described in Section 6.1 shows the performance of humans at predicting theft in Chicago. The area under this ROC curve is 0.368 (37% accuracy). Somewhat unexpectedly, the workers do worse than random (see Figure 13). Compared to this baseline, almost all of our predictors are significantly more accurate.

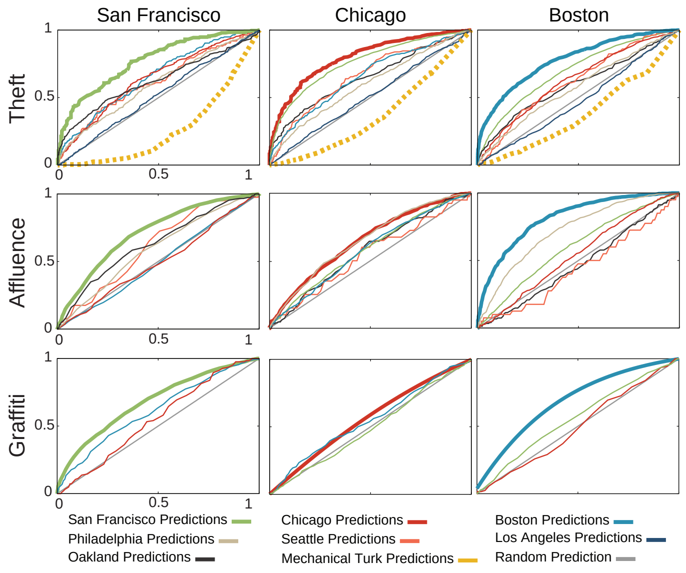


Figure 13: *The ROC curves for the theft, affluence, and graffiti presence statistics. Each row is a statistic, each column is a source city, and each plotted line is an ROC curve for a source city's predictor of a statistic in a target city (colored consistently throughout the plots). Bolded lines correspond to intracity predictions and dotted yellow lines correspond to human predictions. In almost every case our prediction accuracy is significantly better than random and well above the performance of humans in estimating crime in Chicago.*

Our results indicate that humans lack the ability to correctly identify high theft areas based on images This is most likely due to our inability to identify visual elements that are indicative of theft. This makes some intuitive sense since criminals would not operate in areas people avoided because they felt unsafe. Hence it's likely that areas that look safe but aren't are actually the most likely areas for crimes (including theft) to occur.

7 Applications

The results we present in Section 6 show that in many cases we are able to reliably predict thefts, affluence, and graffiti presence in a target city from predictors computed in a different city. To illustrate the applicability of these predictors, we implemented two prototype applications: a statistic-sensitive wayfinding system allowing a user to find directions between locations in a city whilst attempting to avoid or encounter a desired statistic and a system for finding areas in cities where graffiti is likely to be present intended to facilitate graffiti removal efforts.

7.1 A Statistic-Sensitive Wayfinding System

When a person is new to a city or less familiar with a particular city statistic it can be hard to generate routes that either avoid or encounter that statistic. Using our predictors for theft and tree presence, we built a system that allows a user to route him/herself between two points in a city while either avoiding or encountering one of these statistics. Extending the system to other statistics would be a relatively trivial effort.

Given a source and a destination in a city, we compute the shortest path in a weighted graph where the weights are proportional to the distance between nodes, the estimates of a set of chosen statistics of interest, and the weight assigned to those statistics by the user. We

	Target Cities						
Source Cities	San Francisco	Chicago	Boston	Oakland	Los Angeles	Seattle	Philadelphia
San Francisco	0.772	0.639	0.653	0.625	0.516	0.616	0.597
Chicago	0.760	0.798	0.652	0.650	0.512	0.660	0.599
Boston	0.694	0.629	0.779	0.598	0.487	0.621	0.575

Table 2: ***Theft Prediction Performance:*** *The theft statistic generalized relatively well in almost all cities. The performance of the Chicago predictor in San Francisco is our best cross-city accuracy. Los Angeles seemed to be a difficult city to generalize to, but we found that the ground truth data was only recorded at intersections, potentially polluting our results.*

	Target Cities					
Source Cities	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

Table 3: ***Affluence Prediction Performance:*** *The ROC curves for the affluence statistic indicate that it generalizes only in certain circumstances. The Boston predictor generalizes with 72% accuracy in Philadelphia most likely due to their spatial proximity and similar architectural history. For similar reasons the San Francisco predictor generalizes well to Seattle and Oakland. The Chicago predictor shows relatively low performance even in Chicago, although it is just as accurate in Philadelphia implying that improving the predictor's accuracy in Chicago could improve its performance there as well.*

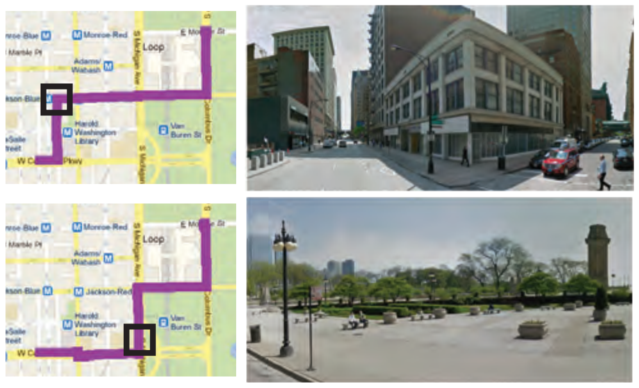


Figure 14: *Visitors to new cities often want to see elements of nature while they explore. In this case, our system has re-routed a user through downtown Chicago, exposing them to more trees along their route. Our statistic-sensitive route is shown on the bottom along with an image of a location along that route that the original route doesn't contain. The top row contains a route optimized for distance only and routes the user through a more urban part of the city.*

use Dijkstra's shortest path algorithm [Dijkstra 1959] to compute the route in realtime as the user adjusts the weights and locations.

To highlight the generalizability of our predictors (analyzed quantitatively in Section 6.2), our wayfinding application uses predictors from a different city than the city we generate the directions in.

Figure 2 shows an example of a route computed that attempts to avoid theft in Chicago based on predictions from a predictor generated in San Francisco. We have plotted the actual instances of crime in Chicago for reference. Our path correctly avoids many of the theft occurrences that would have been encountered if the shortest spatial path were taken.

Our system can also be used to encounter attributes rather than avoid them. It's often desirable to find good walking paths through cities, especially if you're on vacation or simply want to see the more beautiful parts of the city. We use our tree predictor trained in San Francisco to find a path between two points in Chicago. Figure 14 shows an example of a Google StreetView panorama along our computed route as well as one along the original route. The system correctly identifies a path that exhibits a tree coverage. It's worth noting that our computed route is passing by a park but the system is choosing that path automatically, based on images alone.

7.2 User-Assisted Graffiti Locator

Graffiti is a common problem in most cities. One of the challenges that city officials face is simply locating the instances of non-sanctioned graffiti. Typically local governments rely on residents to report instances of illegal graffiti. Unfortunately, a lot of graffiti goes unreported according to our results.

We developed a system that enables a user (such as a government employee) to locate areas in cities that are likely to contain graffiti. Since our visual elements (see Figure 10) are not *directly* related to graffiti, we cannot find the exact region in an image that contains graffiti. However, in most cases we can reliably predict locations in a city where graffiti is likely to occur given examples of where it has occurred before (see Section 6.1).

In Figure 15 we show an example of graffiti in San Francisco that we found within minutes of using our system that was not present in the data we downloaded from their graffiti reporting database. We also show an image of an area where we predict a low chance of graffiti. Note that the two images have a similar visual appearance (relatively flat, gray walls and large, open spaces) and come from locations that are within a block of each other. Despite these two potentially confounding factors, our system correctly identifies the area that contains graffiti and identifies the other region as being graffiti-free.

8 Conclusions and Future Work

We have presented a method for automatically computing predictors capable of estimating the probability of observing a statistic in a city based only on street-level images of the city. We used a set of predictors that we computed for theft, affluence, graffiti presence, and tree presence to analyze the relationship between visual elements and city statistics. Our results indicate that there is a relationship between the visual elements and city statistics, and that the relationship is general enough to predict statistics in new cities. Finally, we presented two applications that use our predictors demonstrating the applicability of this work.

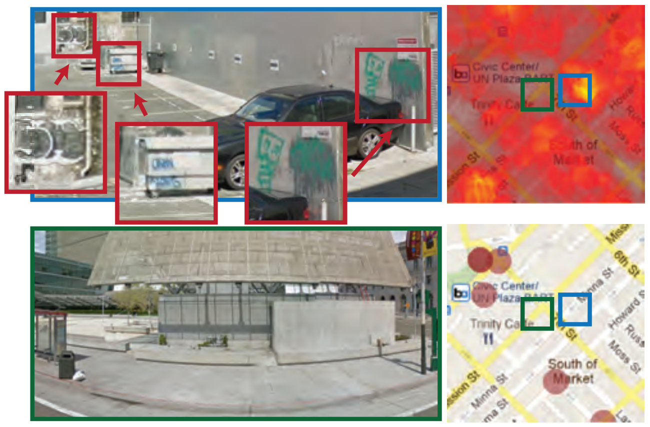


Figure 15: *Using a predictor for graffiti computed from San Francisco data, we are able to find new locations in the city that are likely to contain graffiti. On the right we show our predictions plotted as a heat map (top) versus actual reported instances of graffiti plotted as red circles (bottom). Notice that the region that we predict has graffiti in it was not originally reported.*

We imagine that our predictors could be used in other applications in graphics such as creating visual snapshots of different areas in a city based on a co-occurrence of statistics (e.g. some neighborhoods contain many trees and no graffiti, while others exhibit the opposite) or for rendering a city block with stock 3D models of homes based on our predicted prices. In future work we would like to investigate new ways to extract better models of visual elements that are more discriminative of statistics. In addition, we are interested in the potential applications of our distributed visual processing framework in computer graphics.

Acknowledgements

This work was supported by Nokia and the Intel Science and Technology Center for Visual Computing. We would also like to thank Carl Doesrch and Alyosha Efros for providing their code and their insights.

References

AGARWAL, A., AND TRIGGS, B. 2006. Recovering 3d human pose from monocular images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 28*, 1, 44–58.

BOSER, B. E., GUYON, I. M., AND VAPNIK, V. N. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, ACM, New York, NY, USA, COLT '92, 144–152.

BOYKO, A., AND FUNKHOUSER, T. 2011. Extracting roads from dense point clouds in large scale urban environment. *ISPRS Journal of Photogrammetry and Remote Sensing 66*, 6, S2–S12.

CARLBERG, M., GAO, P., CHEN, G., AND ZAKHOR, A. 2009. Classifying urban landscape in aerial lidar using 3d shape analysis. In *Proceedings of the 16th IEEE international conference on Image processing*, IEEE Press, Piscataway, NJ, USA, ICIP'09, 1681–1684.

CASELLA, G., AND BERGER, R. L. 1990. *Statistical inference*, second ed., vol. 70. Duxbury Press Belmont, CA.

CHANG, C.-C., AND LIN, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2*, 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

COHEN, D., SPEAR, S., SCRIBNER, R., KISSINGER, P., MASON, K., AND WILDGEN, J. 2000. "broken windows" and the risk of gonorrhea. *American Journal of Public Health 90*, 2, 230.

DALAL, N., AND TRIGGS, B. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 886–893 vol. 1.

DEAN, J., AND GHEMAWAT, S. 2008. Mapreduce: simplified data processing on large clusters. *Communications of the ACM 51*, 1, 107–113.

DIJKSTRA, E. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 1, 269–271.

DOERSCH, C., SINGH, S., GUPTA, A., SIVIC, J., AND EFROS, A. A. 2012. What makes paris look like paris? *ACM Transactions on Graphics (SIGGRAPH) 31*, 4.

ELLAWAY, A., MACINTYRE, S., AND BONNEFOY, X. 2005. Graffiti, greenery, and obesity in adults: secondary analysis of european cross sectional survey. *BMJ: British Medical Journal 331*, 7517, 611.

FELZENSZWALB, P., MCALLESTER, D., AND RAMANAN, D. 2008. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 1 –8.

HANLEY, J. A., AND MCNEIL, B. J. 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology 143*, 1, 29–36.

HARA, K., LE, V., AND FROEHLICH, J. 2013. Combining crowdsourcing and google street view to identify street-level accessibility problems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '13, 631–640.

HU, W., TAN, T., WANG, L., AND MAYBANK, S. 2004. 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.

KOLLER, D., WEBER, J., AND MALIK, J. 1994. Robust multiple car tracking with occlusion reasoning. In *Computer Vision ECCV '94*, J.-O. Eklundh, Ed., vol. 800 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 189–196.

KUMAR, N., BERG, A. C., BELHUMEUR, P. N., AND NAYAR, S. K. 2009. Attribute and simile classifiers for face verification. In *Computer Vision, 2009 IEEE 12th International Conference on*, IEEE, 365–372.

LATKIN, C. A., AND CURRY, A. D. 2003. Stressful neighborhoods and depression: a prospective study of the impact of neighborhood disorder. *Journal of Health and Social Behavior*, 34–44.

MAIRAL, J., ELAD, M., AND SAPIRO, G. 2008. Sparse representation for color image restoration. *Image Processing, IEEE Transactions on 17*, 1, 53–69.

MCKENNA, S. J., JABRI, S., DURIC, Z., ROSENFELD, A., AND WECHSLER, H. 2000. Tracking groups of people. *Computer Vision and Image Understanding 80*, 1, 42 – 56.

ROTTENSTEINER, F., AND BRIESE, C. 2002. 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.

RUNDLE, A. G., BADER, M. D., RICHARDS, C. A., NECKERMAN, K. M., AND TEITLER, J. O. 2011. Using google street view to audit neighborhood environments. *American journal of preventive medicine 40*, 1, 94–100.

SECORD, J., AND ZAKHOR, A. 2007. Tree detection in urban regions using aerial lidar and image data. *Geoscience and Remote Sensing Letters, IEEE 4*, 2 (april), 196 –200.

SINGH, S., GUPTA, A., AND EFROS, A. A. 2012. Unsupervised discovery of mid-level discriminative patches. In *Computer Vision–ECCV 2012*. Springer, 73–86.

SMOLA, A., AND SCHLKOPF, B. 2004. A tutorial on support vector regression. *Statistics and Computing 14*, 3, 199–222.

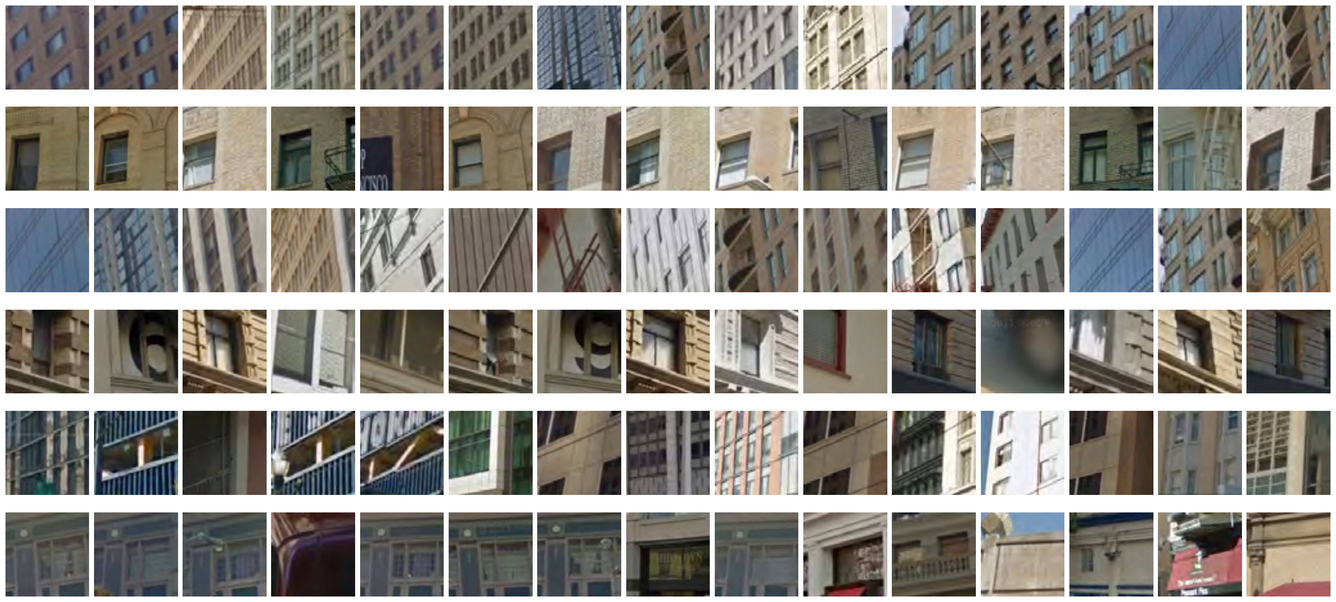
SRINIVASAN, S., LATCHMAN, H., SHEA, J., WONG, T., AND MCNAIR, J. 2004. Airborne traffic surveillance systems: video surveillance of highway traffic. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, ACM, 131–135.

WHITE, T. 2012. *Hadoop: The definitive guide*, third ed. O'Reilly Media, Inc.

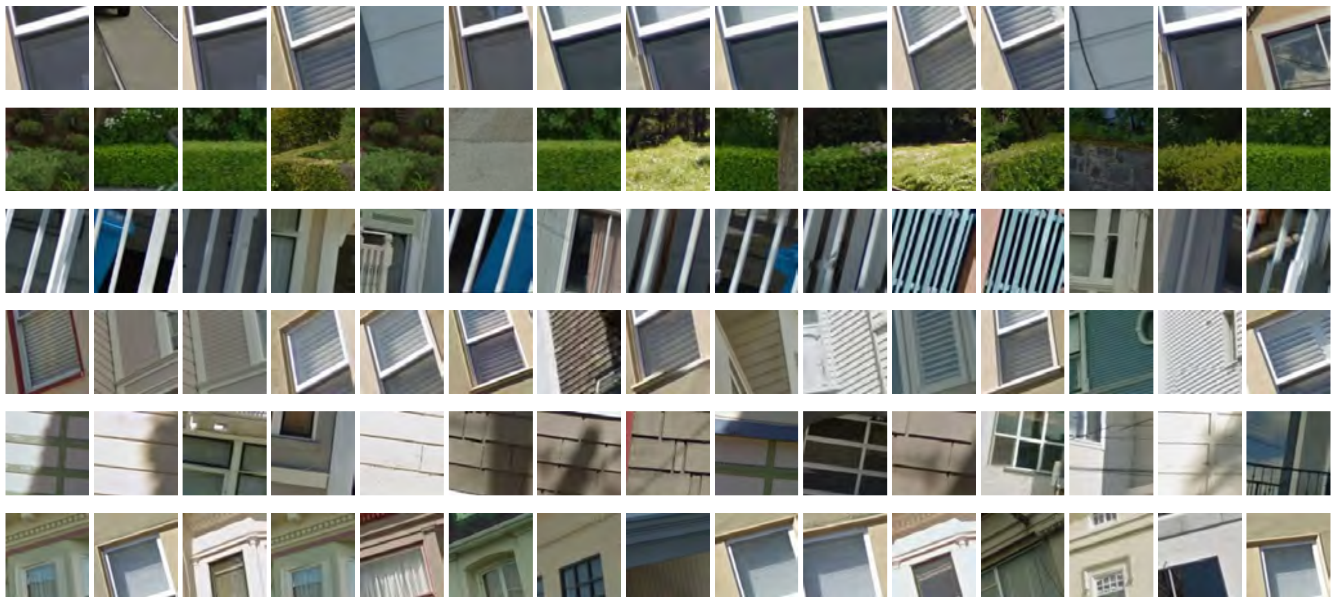
WILSON, J. Q., AND KELLING, G. L. 1982. Broken windows. *Atlantic monthly 249*, 3, 29–38.

ZIMBARDO, P. G. 1969. The human choice: Individuation, reason, and order versus deindividuation, impulse, and chaos. In *Nebraska symposium on motivation*, University of Nebraska Press.

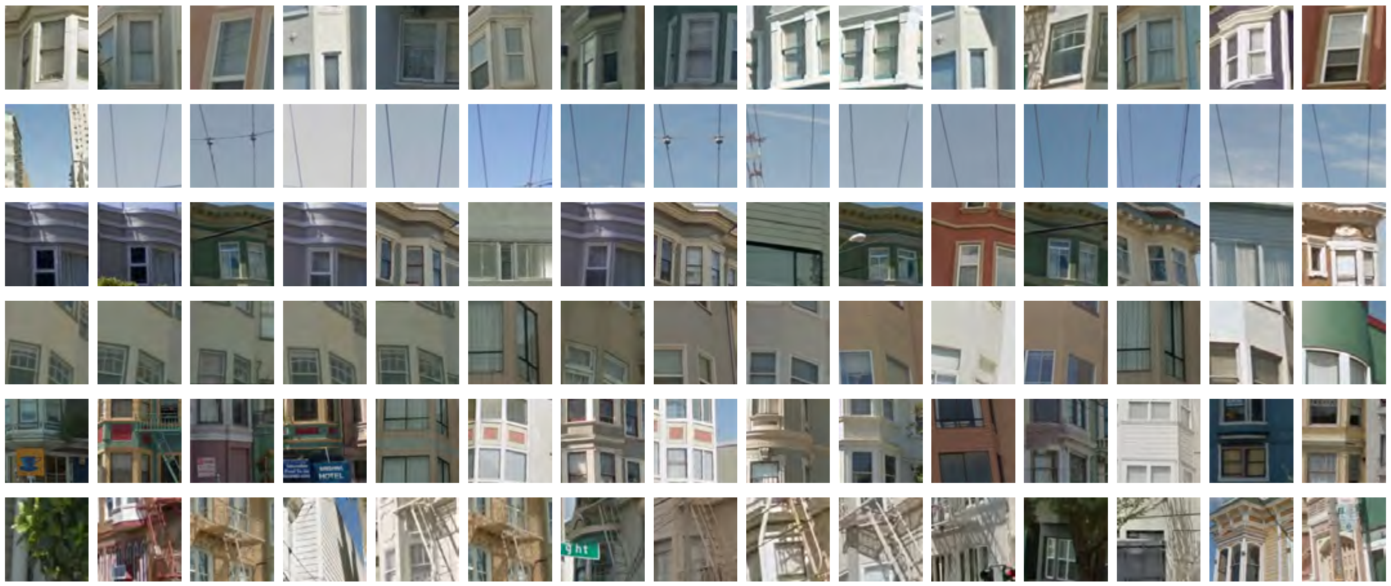
On Relating Visual Elements to City Statistics
Supplemental Material



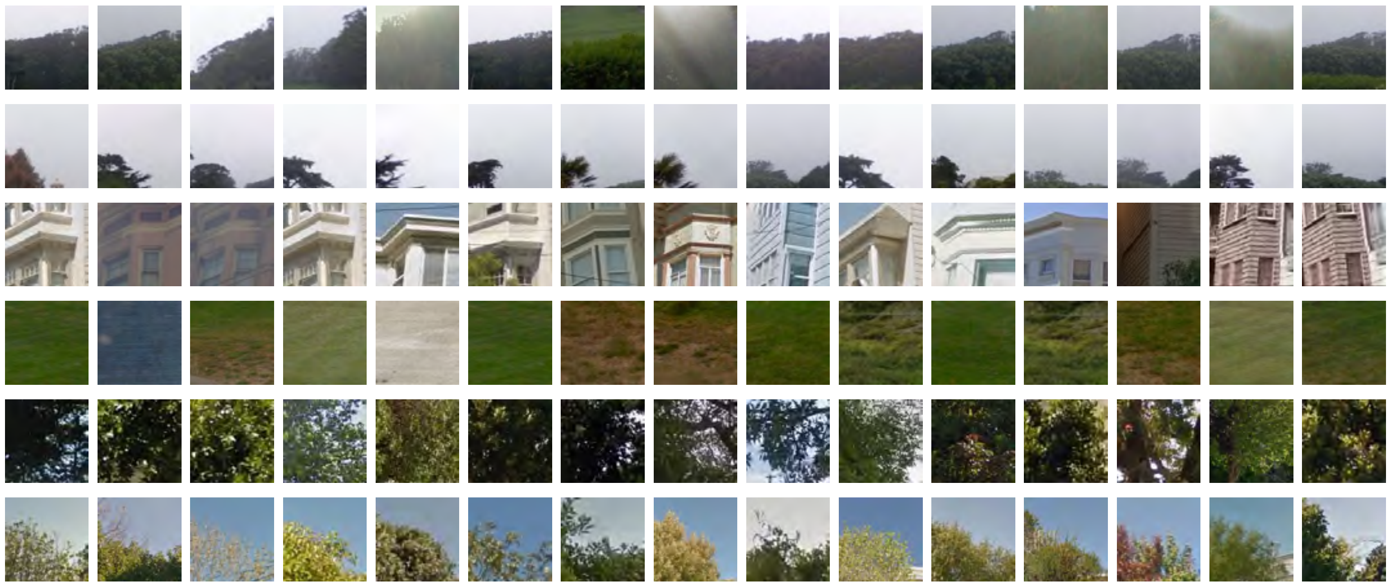
Visual Elements for Theft in San Francisco: *These example visual elements for the theft statistic in San Francisco represent high density windows, windows with fire escapes, and repeated vertical "bars". Most of the thefts in San Francisco occur downtown where these kinds of visual elements are present.*



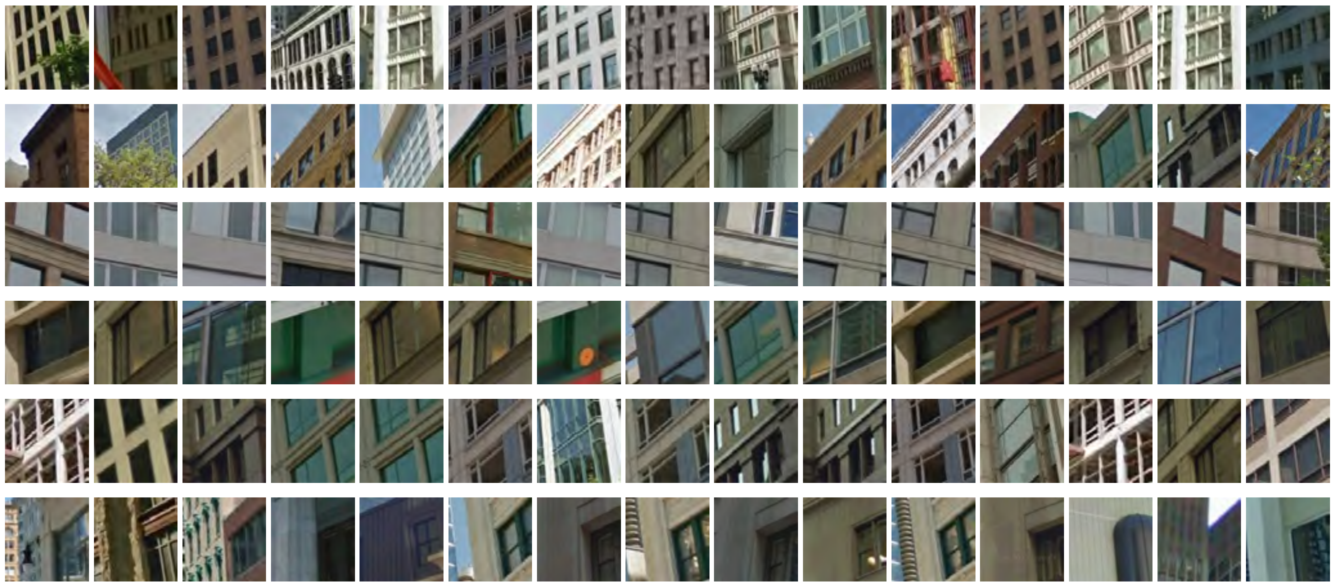
Visual Elements for Affluence in San Francisco: *Affluence in San Francisco is connected to hedges (second row), gates (third row), wood paneling (fifth row), and intricate window moldings (bottom row).*



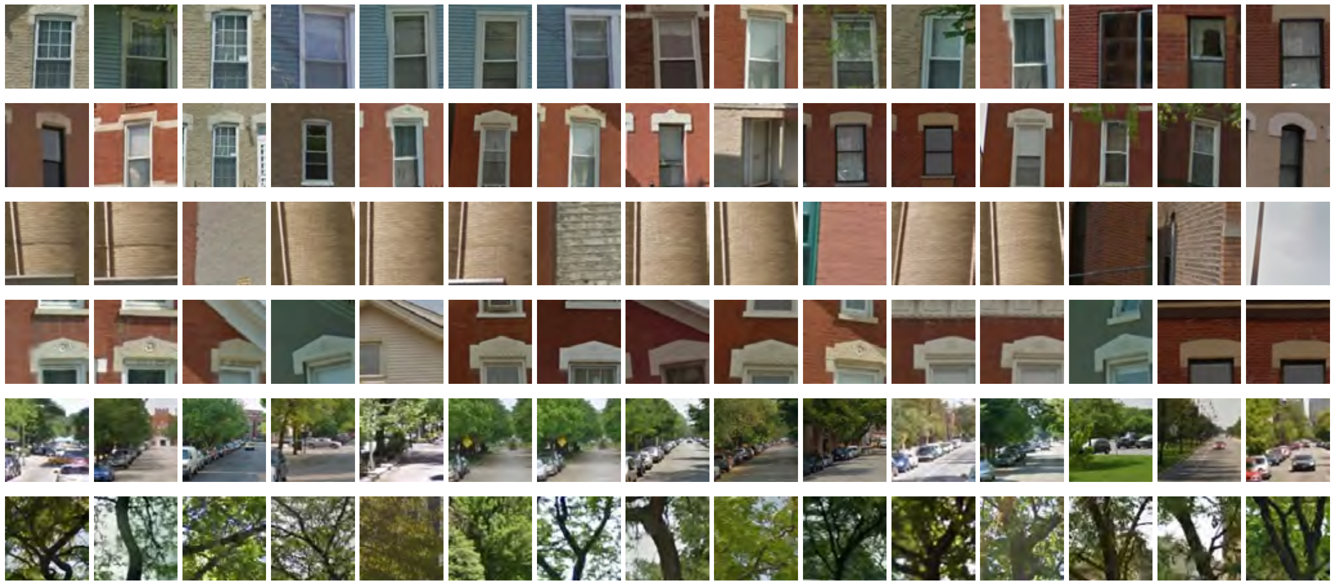
Visual Elements for Graffiti in San Francisco: *Many of these visual elements correspond to architectural styles specific to certain locations in San Francisco (large bayview windows). Fire escapes (bottom row) are also correlated with graffiti.*



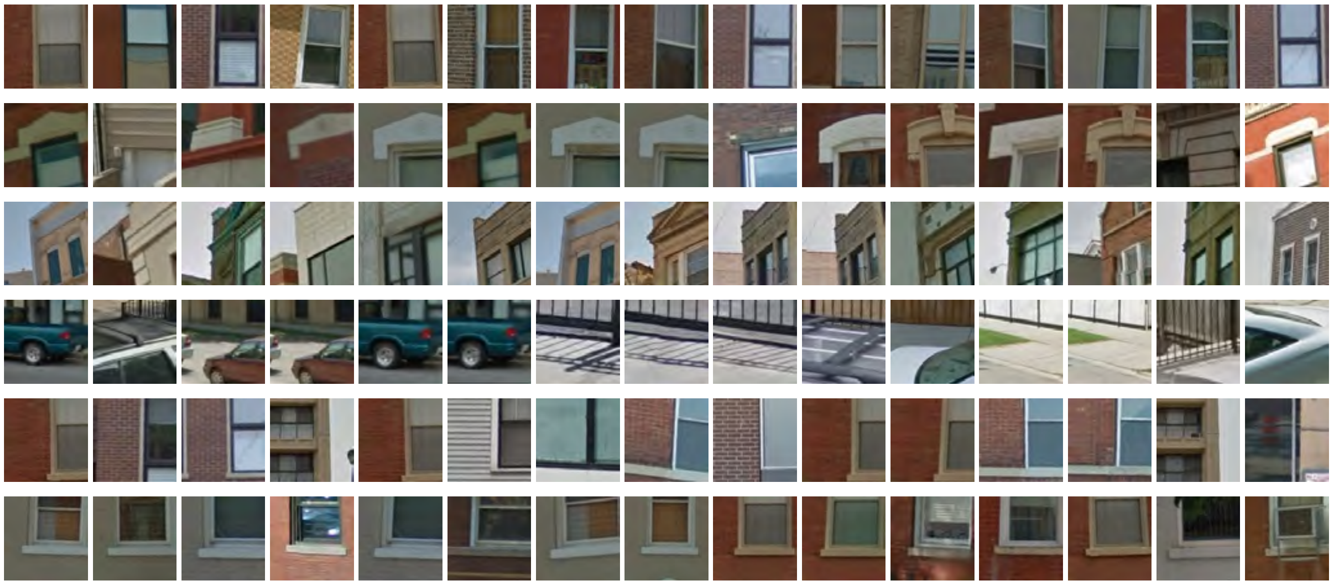
Visual Elements for Trees in San Francisco: *Most of the visual elements for trees in San Francisco are simply examples of different styles of trees. One interesting note is that the third row is a visual element of an architectural style, likely specific to the presence of trees in San Francisco.*



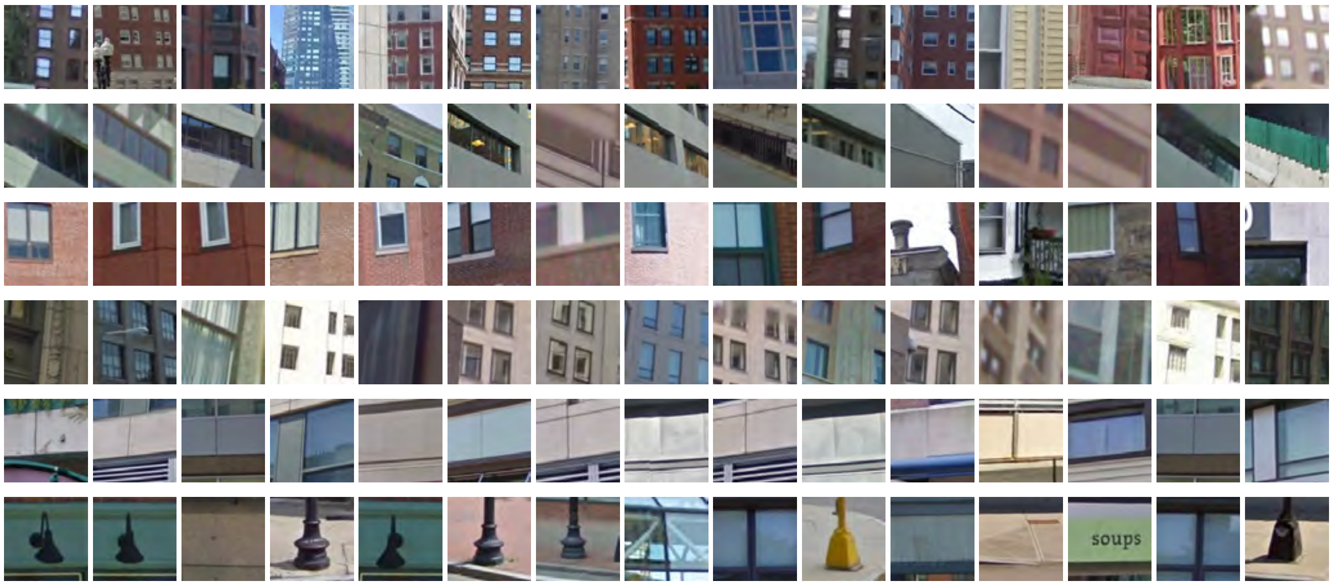
Visual Elements for Theft in Chicago: *Similar to San Francisco, the visual elements in Chicago relate to high density windows and repeated vertical bars. This similarity is likely the cause for the strong generalizability between these two cities for the theft predictors.*



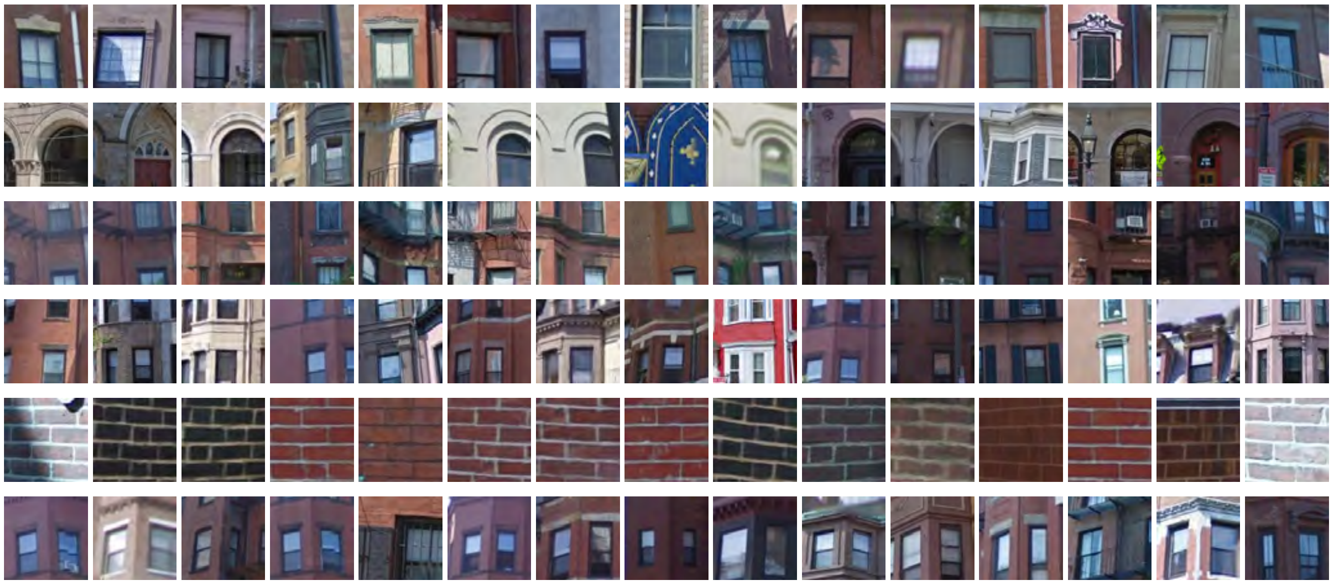
Visual Elements for Affluence in Chicago: *Many of the visual elements for affluence in Chicago relate to the moldings around windows and the presence of trees. The fourth row shows the classic "tree-lined street" visual element that we often associate with wealth.*



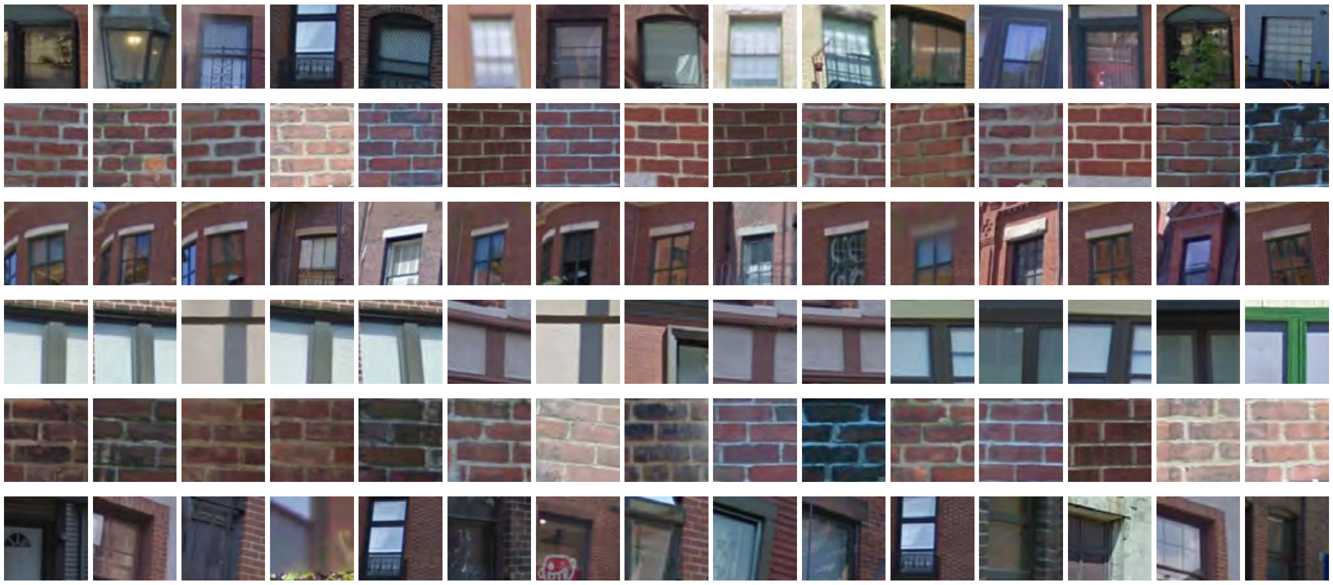
Visual Elements for Graffiti in Chicago: *Visual elements like simplified windows (first, fourth, and fifth rows) and the specific architectural style from the second row seem like they bear a reasonable relation to graffiti. However, we found that graffiti in Chicago was difficult to predict indicating that these visual elements may not be discriminative enough.*



Visual Elements for Theft in Boston: *Just as in all of our other cities we note visual elements related to theft that exhibit high density windows. In addition, the bottom row seems to be a visual element for lamp posts. We believe that this visual element is important because lamp posts are often present in areas of high density foot traffic, which are prone to theft.*



Visual Elements for Affluence in Boston: *Our predictor for affluence in Boston performed very well in our experiments. We attribute this performance to the variation in the visual elements and their individual uniformity (all of the images in a row look visually similar). Rounded door and window thresholds (second row), bayview-style windows (fourth and last rows), and darker-colored bricks (fifth row) all make intuitive sense as visual elements that relate to affluence.*



Visual Elements for Graffiti in Boston: *The visual elements that relate to graffiti in Boston include windows with bars or other obstructions (first row), bricks that seem unkempt (second and fifth rows), and particular styles of windows. The last row actually contains an image of graffiti (seventh column), although it is unclear how this particular visual element is related to it.*