Learning Deep Representations for Ground-to-Aerial Geolocalization

Tsung-Yi Lin[†] tl483@cornell.edu
Yin Cui[†] yc984@cornell.edu
Serge Belongie[†] sjb344@cornell.edu
James Hays[§] hays@cs.brown.edu
[†]Cornell Tech [§]Brown University

Abstract

The recent availability of geo-tagged images and rich geospatial data has inspired a number of algorithms for image based geolocalization. Most approaches predict the location of a query image by matching to ground-level images with known locations (e.g., street-view data). However, most of the Earth does not have ground-level reference photos available. Fortunately, more complete coverage is provided by oblique aerial or "bird's eye" imagery. In this work, we localize a ground-level query image by matching it to a reference database of aerial imagery. We use publicly available data to build a dataset of $78K$ aligned cross-view image pairs. The primary challenge for this task is that traditional computer vision approaches cannot handle the wide baseline and appearance variation of these cross-view pairs. We use our dataset to learn a feature representation in which matching views are near one another and mismatched views are far apart. Our proposed approach, Where-CNN, is inspired by deep learning success in face verification and achieves significant improvements over traditional hand-crafted features and existing deep features learned from other large-scale databases. We show the effectiveness of Where-CNN in finding matches between street view and aerial view imagery and demonstrate the ability of our learned features to generalize to novel locations.

1. Introduction

Consider the photo on the left side of Fig. 1. How can we estimate where it was taken? Most existing methods predict image location via matching to other ground-level photos with known locations, but what if that data isn't available? In this work, we present a method to match ground level queries to aerial imagery. The right side of Fig. 1 shows one such aerial image out of thousands in our database. Matching across these disparate visual domains is difficult for two main reasons. Geometrically, the wide baseline induces a large amount of occlusion in each view (e.g., we only see building roofs in aerial views, and occlusions by trees and street parking are common in street-level views). Furthermore, the photos may have been captured at different times with different lighting, weather, and season. The main purpose of this paper is to investigate the feasibility of ground to aerial matching in light of these challenges.

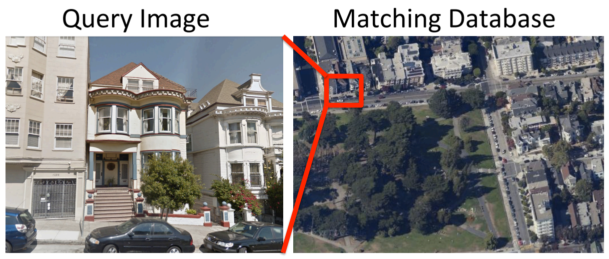


Figure 1: Given a query street-view image, this paper aims to find where it was taken by matching it to a city-scale aerial view image database.

In this paper, we frame photo geolocalization as an identity verification task where only one correct location exists in a city-scale region. This is analogous to the well-studied face verification [18] task in which algorithms must decide whether a pair of input photos depict the same individual. Recent methods achieved high performances by extracting hand-crafted features at aligned fiducial points [5, 6, 8]. While these approaches are fairly specific to the face domain, DeepFace [25] instead achieved impressive accuracy by learning a deep feature representation on aligned face images with massive additional training data. Inspired by the success of DeepFace, we first create a large-scale dataset that contains cross-view images aligned by publicly available coarse depth estimates on ground images. Then, a low dimensional feature representation is learned by a deep "Siamese network" [10] with the objective that the cross-view image pairs of the same location will be close while pairs of different locations or views will be far away.

The contributions of this paper are three-fold: (1) our method can localize a photo without using ground-level reference imagery by matching to aerial imagery. (2) We present a novel method to create a large-scale cross-view training dataset from public data sources. (3) We examine traditional computer vision features and several recent deep learning strategies in novel cross-domain learning task.

Image Geolocalization Methods. IM2GPS [17] was an early, influential approach to predict image location by matching visual appearance. With the increasing number of geo-tagged images from photo sharing websites and tremendous effort by Google to capture the world at street level [1], many geolocalization techniques boil down to fast nearest neighbor search over a large ground-level image database [9, 21, 29]. Despite the impressive scale of community geo-tagged photo collections and street-view databases, most of the Earth still has no ground-level reference imagery. To address this, recent methods localize query images by matching them to digital elevation maps of mountainous terrain [2] or cities [3].

The three most similar methods to our paper are recent "cross-view" techniques which also match ground-level photos to overhead or aerial imagery. Lin et al. [22] match ground-level queries to other ground-level reference photos as in traditional geolocalization, but then use the overhead appearance and land cover attributes of those ground-level matches to build sliding-window classifiers in the aerial and land cover domain. A limitation of this approach is that there is no direct comparison across views and thus it cannot latch onto particular discriminative elements of the query – it can only build a classifier for the types of elements seen in similar scenes.

Like our approach, Bansal et al. [4] investigate ultra-wide baseline matching between street-view images and 45° aerial view images. They propose self-similarity descriptors for large building facades which are distinctive enough to be matched across the wide cross-view baseline. They demonstrate cross-view matching for buildings in a urban region. In this paper, we *learn* a more universal cross-view representation and evaluate our system at the scale of entire cities that includes both urban and suburban regions.

Shan et al. [24] propose ground-aerial image matching that first builds 3D point cloud from ground images with noisy geo-tags and performs depth-based warping to reproject ground images to one aerial view based on the estimated geometry. The warped ground images then match to aerial imagery by sparse keypoints matching. To reduce search space, the aerial imagery is cropped based on the rough geolocation associated with ground images. The method shows promising results on referencing landmark images to aerial imagery with pixel-level accuracy. Both [24] and this paper share the similar procedure to align ground and aerial images as a preprocessing step. Moreover, we learn a novel feature representation that is capable to search all possible aerial matches in a city.

2. Dataset

For the experiments in this paper, we collect Google street-view and 45° aerial view images from seven cities

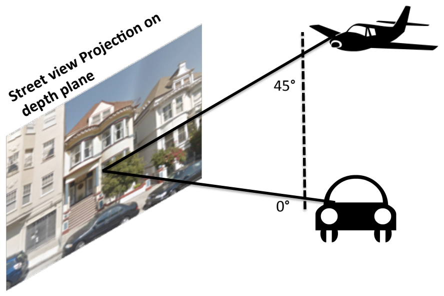


Figure 2: This diagram shows the relationship between our cross-view pairs sampled from aerial images and street-view panoramas.

– San Francisco, San Diego, Chicago, and Charleston in the United States as well as Tokyo, Rome, and Lyon to test how well our method generalizes to novel locations. The data includes both urban and suburban areas for each city.

Before we attempt cross-view matching, a major challenge is simply to establish ground-truth correspondences between street-level and aerial views. This is challenging due to unknown geolocation, scale, and surface orientation in the raw images. For our study, we show that pairs of coarsely aligned cross-view images can be generated with the depth estimates and meta data of street-view images provided by Google. First, we project 2D street-view image to 3D world coordinates system by leveraging the heading direction of street-view car and pixel-wise coarse depth planes from Anguelov et al. [1]. Second, we assume an orthographic camera model for the aerial view with viewing directions aligned with four cardinal directions (north, east, south, and west) and tilted 45° downward. Finally, the scale of street-view depth estimates and aerial view imagery is calibrated so the street-view images can be reprojected onto aerial view image plane through the street-view depth estimates.

For our experiments, we generate cross-view image pairs with area of 15×15 meters (256×256 pixels). Each panorama image can at most contribute two cross-view pairs, e.g., a panorama image captured by an east-facing street-view car can generate north and south-facing cross-view image pairs. Each cropped street-view image is centered at 0° tilt angle and aligned with a cardinal direction. For simplicity, we project the street-view crop onto a single depth plane at the center pixel of an image. Fig. 2 shows the relationship of our cross-view pairs. In our experiments, we only generate cross-view image pairs if a depth estimate exists at the center of street-view crop.

Fig. 3 shows examples of cross-view image pairs generated by our method. Although these pairs are roughly aligned, the error in geolocation of ground and aerial images and depth estimates lead to differences in scale, shear,

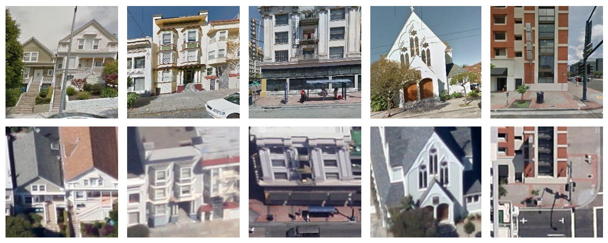


Figure 3: Corresponding pairs of street-view (top) and 45° aerial view (bottom) images aligned using the publicly available depth estimates.

translation, and projective distortion. The images are captured from different imaging devices at dramatically different distances and thus exhibit different point spread functions and color calibrations. Finally, occlusions add to the appearance disparities (e.g., a roof can only be seen from aerial view and a tree occludes different parts of building from street-view).

Traditional matching methods based on local features extracted from sparse keypoints struggle to match such cross-view image pairs. Fig. 4 shows an ground-aerial image pair from our database, where the left side shows interest points found by SIFT Detector [23] and the right side shows top 10 key points matching. We can see that key points on street view image mainly focus on the bottom part of the building and another tall building behind it which is not even visible in the aerial view image. In our initial experiments, keypoint matching methods (even with RANSAC to filter outliers) do not work across these views, whereas the proposed method can correctly match them (Fig. 8a).

3. Cross-view Image Matching

We frame our problem as an identity verification task when designing and training our deep cross-view network. However, unlike classic verification tasks such as face verification, we do not have category labels for each instance (or, equivalently, every single location is a unique category). At test time, we would ideally have a verification method that reports 'true' for the correct location and 'false' for every other locations. However, we would like an evaluation metric that rewards near-misses (e.g. the correct match is among the top scoring matches). Therefore we evaluate our cross-view geolocalization as an image retrieval problem and report Average Precision for each algorithm.

Our goal is to find a good representation $f(.)$ for cross-view images. $f(.)$ could be either hand-crafted or learned from data. Then, when a novel query ground-view image x comes in, we want to find the matched patch y from aerial view imagery Y accurately and efficiently. The Euclidean Distance is used as the distance metric to find the nearest neighbor $y \in Y$ in feature space as the matched patch

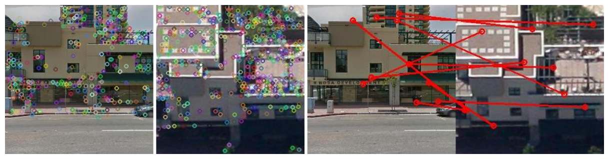


Figure 4: Key points matching fails on aerial-to-ground view image pairs for our problem.

$match(x)$ for x:

$$match(x) = \arg\min_{y \in Y} \|f(x) - f(y)\|_2 \tag{1}$$

3.1. Feature Representations

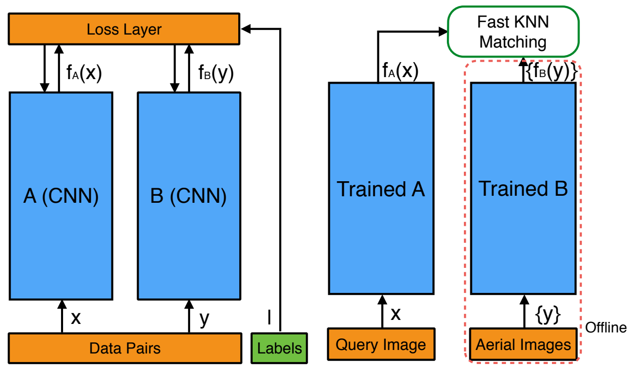
There are no "standard" feature representations for the ultra-wide baseline cross-view matching, because it is a relatively unexplored task. There are promising representations from other domains, though – traditional hand-designed features which have been shown to work well in object detection and recent breakthroughs in deep convolutional networks. Specifically, we focus on three types of feature representations: (1) hand-crafted features; (2) generic deep feature representations; and (3) learned deep feature representations for our data. We discuss each type of feature in details in Sec. 4.3.

3.2. Network Architecture and Loss Function

Inspired by the early "Siamese Network" [10] approach and the more recent DeepFace [25] and Deep Ranking [27] methods, we use a pair-based network structure illustrated in Fig. 5a to learn deep representations from data for distinguishing matched and unmatched cross-view image pairs.

During training, the input to the network is a pair of images $x \in X$ and $y \in Y$, where X and Y are street-view imagery and aerial view imagery in the training set, respectively. The input pair x and y are fed into two deep convolutional neural networks (CNN) A and B , which have same architecture. The goal of our convolutional network is to learn a feature representation (non-linear embedding) $f(\cdot)$ that embed raw input images $x, y \in \mathbb{R}^n$ from different views to a lower dimensional space as $f_A(x), f_B(y) \in \mathbb{R}^d$ where images from matched pairs are pulled closer whereas images from unmatched pairs are pushed far way from each other. Here, n is the number of pixels for input image x or y and $d \ll n$ is the dimension of feature representation $f_A(x)$ or $f_B(y)$.

Note that two CNNs A and B could be either identical with shared parameters or distinct. In the case of sharing parameters, a general deep representation is learned across between street view and aerial view. On the other hand, in the case of different parameters, domain specific deep



(a) Training (b) Testing

Figure 5: Our network architecture for cross-view image matching.

representations are learned by A and B, respectively. For simplicity, we will abuse a single notation $f(.)$ to represent both $f_A(\cdot)$ and $f_B(\cdot)$.

In order to optimize the proposed network, we need to use a loss function that fits our goal. More specifically, we want to let matched pairs have small Euclidean Distance close to 0 and let unmatched pairs have large Euclidean Distance larger than a margin m. Therefore, we used the Contrastive Loss Function proposed in [16] as our loss function, which can be expressed as:

$$\mathcal{L}(x, y, l) = \frac{1}{2} l D^2 + \frac{1}{2}(1 - l) \max(0, (m - D^2)) \quad (2)$$

where $l \in \{0, 1\}$ is the label indicating whether the input pair x, y is a matched pair or not ($l = 1$ if matched, $l = 0$ if unmatched), $m > 0$ is the margin for unmatched pairs and $D = \|f(x) - f(y)\|_2$ is the Euclidean Distance between $f(x)$ and $f(y)$ in feature space.

The loss function in Eqn. 2 penalizes matched pairs by the squared Euclidean distances and mismatched pairs by the squared differences of the distances to the margin m for the distances that are smaller than m. Minimizing the loss function in Eqn. 2 pulls matched pairs closer and pushes mismatched pairs far away. After learning the parameters of the deep network which produces our feature representation $f(\cdot)$, we can pre-compute $f(y)$ offline for all $y \in Y$, where Y is our aerial imagery. Then, for a given query x during test time, we can compute $f(x)$ by only one forward pass on the learned deep network A and find nearest-neighbors from $f(Y)$ as expressed in Eqn. 1. The precomputation of $f(y)$ enables us to use fast algorithms such as Locality Sensitive Hashing [15, 7] in finding K nearest-neighbors. Thus our method can perform cross-view matching at city scale in real-time. The test stage is illustrated in Fig. 5b.

3.3. Learning Feature Embedding

Our Siamese network is composed of two identical CNNs modified from [20]. The last fully connected layer fc8 is stripped off and the $4,096$ dimensions from the second last fully connected layer fc7 are used as the feature representation. We fine-tune a pre-trained model by setting the learning rate 10^{-5} for fc7 and 10^{-7} for other layers. In Sec. 4.3, we discuss more details on the selection of pre-trained models. On top of fc7, an additional mean-variance normalization layer is added to normalize features to zero mean and unit standard deviation. The normalization layer prevents features from arbitrarily scaling during training. Because the scale of feature is fixed, it is easy to select the margin for Contrastive Loss. We set the margin to be the average squared pair distance over training data. The smaller the margin, the more that the learning is influenced by "hard negatives". In our case, only pairs of data with distances smaller than the average are used to compute gradients to update the network parameters.

4. Experiments

Our experiments compare the effectiveness of the feature representation learned from our database using a deep convolutional network against traditional hand-crafted features and deep features not optimized for our problem. In addition, we measure how well a representation learned on some cities generalizes to testing on an unseen city.

4.1. Experiments Setup

Training and Test Sets. As described in Sec. 2, we collected 78k pairs of Google street-view images and their corresponding aerial images. Those image pairs are partitioned into training set, which is used to train our deep neural networks, and test set, which is used to validate the effectiveness of the learned features.

We divide our collected image pairs into training and test sets based on the cardinal viewing direction (azimuth). Specifically, we use image pairs that have viewing directions of $0°$, $90°$ and $270°$ as training data. Image pairs that have viewing direction of $180°$ are used as test data. Under this partition, all image pairs in the training set have viewing directions either orthogonal or opposite to the image pairs in our test set, which minimizes the overlapping of image content across training and testing. To test the ability of learned representations to generalize to a novel location, we hold-out Tokyo, Rome, and Lyon for experiments in Sec. 5.

We end up with $37.5K$ matched pairs in training set and $12.5K$ matched pairs in test set and hold out images from San Francisco, Chicago, San Diego, and Charleston. We generate $20\times$ as many *unmatched* pairs randomly for both training and test. Together, the matched and unmatched pairs total $0.8M$ in training set and $0.26M$ pairs in test set. In both training and testing, $1/21$ of image pairs are matched pairs and $20/21$ are unmatched pairs.

Deep Convolutional Networks. For the training of the proposed deep convolutional network described in Sec. 3.2, we used all $0.8M$ image pairs from our training set (the ratio of matched and unmatched pairs is $1 : 20$). Similar to the case of Deep Face [25] and Deep Ranking [27], our goal is to learn deep representations that capture the similarity between images. However, as argued in Sec. 1 and Sec. 2, our database is more challenging than Deep Face and Deep Ranking and it is hard to build a discriminative representation from scratch. Therefore, we use the parameters from deep networks pre-trained on large-scale object and scene databases [20, 30] as the initialization to train our model.

Our model was trained using the publicly available Caffe implementation [19] on a Nvidia Grid K520 GPU. It took about 4 days to finish $250,000$ iterations of training, which is approximately 50 epochs of the training set.

4.2. Performance Evaluation Metric

As we mentioned in Sec. 3, we evaluate our problem as image retrieval problem. There are many ways to evaluate the performance of retrieval, such as top K hits, Average Precision at K, Receiver Operator Characteristic (ROC) curve or Precision-Recall (PR) curve. Among them, we present PR curves because they often shows a more informative picture of a retrieval method's performance compared with other evaluation metrics [12]. We also report Average Precision (AP), the area under PR curve, as the quantitative evaluation metric.

4.3. Feature Representations and Learning

We compare the performance of the following feature representations:

HOG 2x2. The histogram of oriented gradients (HOG) descriptor is widely used for object detection and scene classification [11, 14, 28]. We extract HOG features in an 8×8 grid and then concatenate 2x2 neighboring HOG blocks after normalization to build 52 dimensional local descriptors without spatial pyramid. These local features are quantized into a bag-of-words representation with 300 visual words learned from k-means clustering on the training set.

ImageNet-CNN feature and Places-CNN features. We extract features from deep convolutional networks pre-trained for classification on large-scale object and scene databases [20, 30]. Specifically, ImageNet-CNN is the AlexNet [20] trained on ImageNet database and Places-CNN is the AlexNet trained on Places database. Since it has been shown that deep convolutional networks learned from large-scale databases can extract generic feature representations that generalize well on other databases [13, 30], we directly extracted $4,096$ dimensional output from the fc7 layer of the ImageNet-CNN and Places-CNN as the feature for cross-view matching.

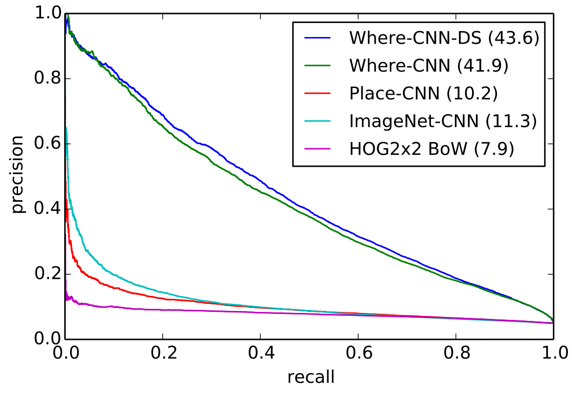


Figure 6: PR curves and corresponding APs for different feature representations.

Where-CNN(-DS) feature. We extract features from the deep convolutional network illustrated in Fig. 5 trained on our database. Where-CNN is domain-independent feature extractor trained with shared parameters between A and B, and Where-CNN-DS is domain-specific feature extractor trained without sharing parameters. We train our deep network using publicly available Caffe implementation [19] on our training set. We initialize the parameters for Where-CNN in the training stage with the learned parameters from both pre-trained ImageNet-CNN and Places-CNN. The comparison between different initialization strategies is shown in Sec. 4.4. After training on the cross-view image pairs, we learn a $4,096$ dimension common feature embedding from fc7 layer of network A and B.

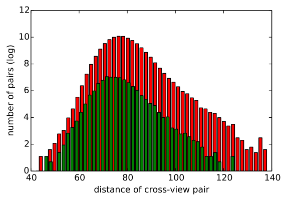
4.4. Quantitative Results and Analysis

In this section, we aim to verify the effectiveness of the deep feature learned by Where-CNN. Fig. 6 plots the PR curves and corresponding APs for the three classes of feature representations examined. From this we make the following observations: (1) Features extracted from the proposed Where-CNN and Where-CNN-DS achieve the best performance by a significant margin compared with hand-crafted features and ImageNet-CNN and Places-CNN features. The architecture of Where-CNN lets us learn a good representation which makes cross-view matched pairs closer than unmatched pairs. (2) Features extracted from Places-CNN and ImageNet-CNN achieve similar performance but both outperform hand-crafted feature by a large margin. (3) The domain-specific model (Where-CNN-DS) performs slightly better than domain-indepedent model (Where-CNN).

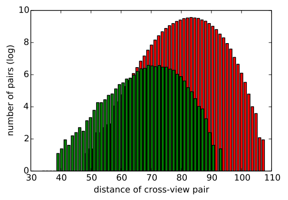
We also compare the effect of different initialization strategies in Table 1. The Where-CNN trained with param-

Table 1: Comparison between different initialization.

Where-CNN	ImageNet init.	Places init.
AP	41.9%	41.4%

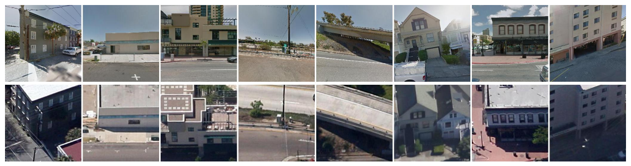


(a) ImageNet-CNN feature.

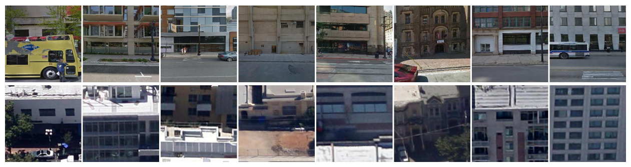


(b) Where-CNN feature.

Figure 7: Histogram of pairwise distances of features from (a) ImageNet-CNN and (b) Where-CNN on test set.



(a) Easy positive pairs.



(b) Hard negative pairs.

Figure 8: (a) The most similar true positive matches and (b) the most similar false positive matches. For each, the first row shows street-view images and the second row shows corresponding aerial images.

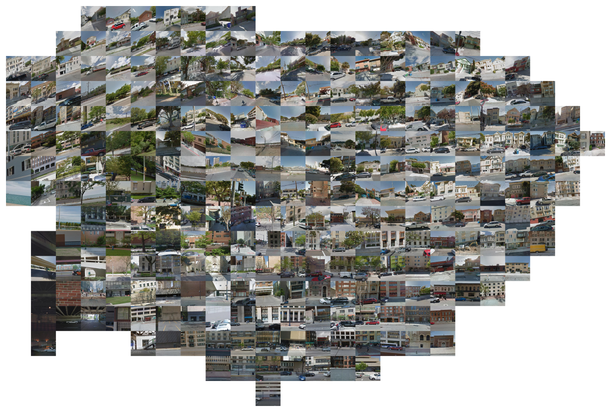


Figure 9: Two dimensional feature embedding. The image are grouped by architecture type and orientation.

eters initialized from ImageNet-CNN achieves similar performance as when initialized from Places-CNN, which indicates our model is robust with different initialization parameters as long as the parameters come from a model trained on large-scale databases that can provide generic representations.

To demonstrate what has been learned by Where-CNN, we compute the histogram of pairwise Euclidean distances of ImageNet-CNN and Where-CNN on the test set in Fig. 7. The green bars represent pairwise distances of matched pairs and red bars represent pairwise distances of unmatched pairs. The pair distance distribution of ImageNet-CNN shows the initial distance distribution of Where-CNN without learning. Obviously, the training process of Where-CNN on the cross-view image pairs effectively pulls matched pairs together and pushes unmatched pairs away.

For additional qualitative evaluation we show easy positives and hard negatives encountered by Where-CNN on our test set in Fig. 8. Easy positives are examples from top 100 true positives (correctly matched pairs with top 100 smallest distances) returned on test set by using Where-CNN feature. Similarly, hard negatives are examples from top 100 false positives (mismatched pairs with top 100 smallest distances). We can see Where-CNN is able to find correct matches from different views even they have small shared regions. For the hard negatives, although they are all mismatched pairs, they look similar to each other and often share structural patterns.

4.5. Feature Visualization

In this section, we try to shed light on why deep features are more capable for cross-view matching by visualizating the feature embedding and showing images that activate particular units at the output feature layer. In Fig. 9, we show an image grid that represents a 2 dimensional embedding of Where-CNN features for street-view images. The embedding is computed by t-SNE [26]. The visualization reveals two influential factors in the feature: (1) architecture type; (2) orientation. For examples, the images at the bottom right show buildings with repetitive windows that look like office buildings and the top right shows two-story residential buildings. The top left part of embedding captures the orientation of images. It is not surprising that orientation is a strong visual cue to learn; however, it is not desired since the orientation of image encodes its relative direction to cardinal direction which might not be known at the test time. Fortunately, we find that given the same orientation, the images with different architecture types are still grouped at different locations in the embedding. This means the unknown orientation at the test time will be one dimension we need to sweep through from -45° to 45° to match the aerial view database.

In addition to looking at the entire feature embedding space, we also examine single unit activation at the output

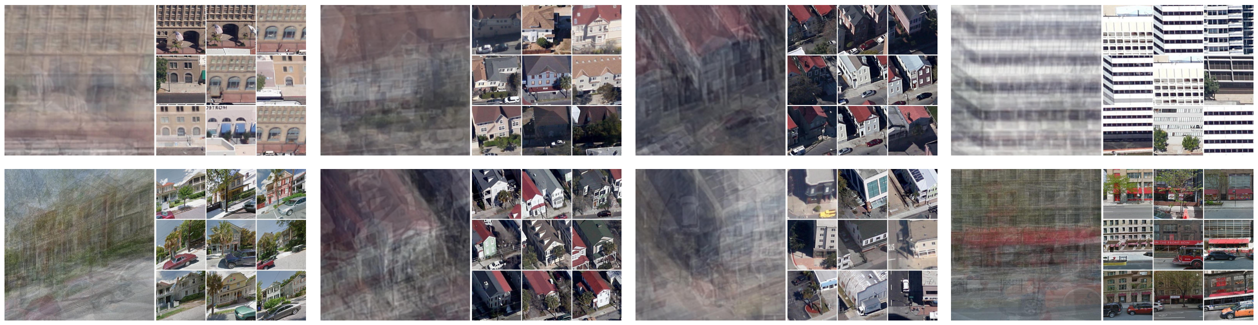


Figure 10: Images that produce strong activations of particular units at the output feature layer.

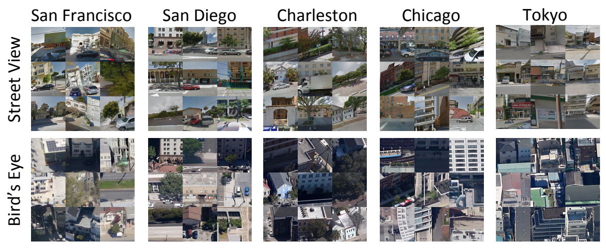


Figure 11: Samples of corresponding cross-view pairs in 5 sampled city.

feature layer. Fig. 10 shows the average images and top 9 images that activate a certain unit most strongly at the output feature layer. Given a unit, ideally it will fire for images with similar structural patterns in street-view or aerial view domain. From Fig. 10, we can identify units for "office building with repetitive windows", "house with triangular roof" or "building with arch doors" that are activated by the corresponding architecture types in street-view or aerial view domains.

5. Geolocalization

In this section, we demonstrate the performance of Where-CNN-DS on a geolocalization task in which one street-view query image is localized by matching against $\sim$10,000 aerial images in each city. For simplicity, we assume the orientation and depth of street view query images are known. In a real world scenario, our method requires either depth estimation from machine or human and a sweep through possible surface normal from -45° to 45°.

We test Where-CNN-DS on 7 cities: San Francisco, San Diego, Chicago, Charleston, Rome, Lyon, and Tokyo. The images in Rome, Lyon, and Tokyo are completely held-out from training data so we can see if Where-CNN-DS can generalize to unseen cities. Fig. 11 shows a snapshot of corresponding pairs of street-view and aerial images randomly drawn for 5 sampled city. It's in particularly challenging for Chicago and Tokyo where tall, crowded buildings severely occlude each other in aerial view.

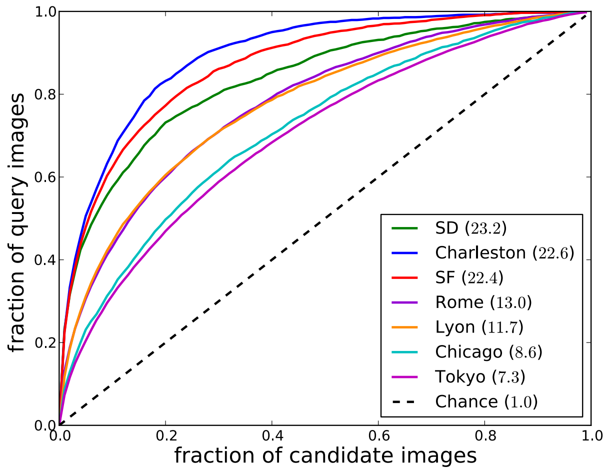


Figure 12: Geolocalization accuracy in various cities. The x-axis is the number of k nearest neighbors considered and the y-axis is the number of queries for which a successful match is within the top k.

Fig. 12 shows the fraction of queries correctly localized as a function of the number of candidates considered. As in Lin et al. [22], we focus on the frequency of test cases for which the correct location was among the top 1% of retrieved candidates (or equivalently, the geolocation estimate has been correctly narrowed to 1% of the search area). Under this criteria, we correctly localize over 22% of queries in Charleston, San Francisco, and San Diego. The accuracy in Chicago and Tokyo are much lower at 8.6% and 7.3%, respectively. Note that our task is much finer scale (15×15 meters) than [22] (180×180 meters).

To gain more insight, Fig. 13 shows examples of query images, the top 12 matched aerial images, and the heat map that indicates possible locations. The probability goes from low to high as the color changes from blue to green to yellow to red. The top two rows show examples where the aerial image with the lowest distance is correctly matched.

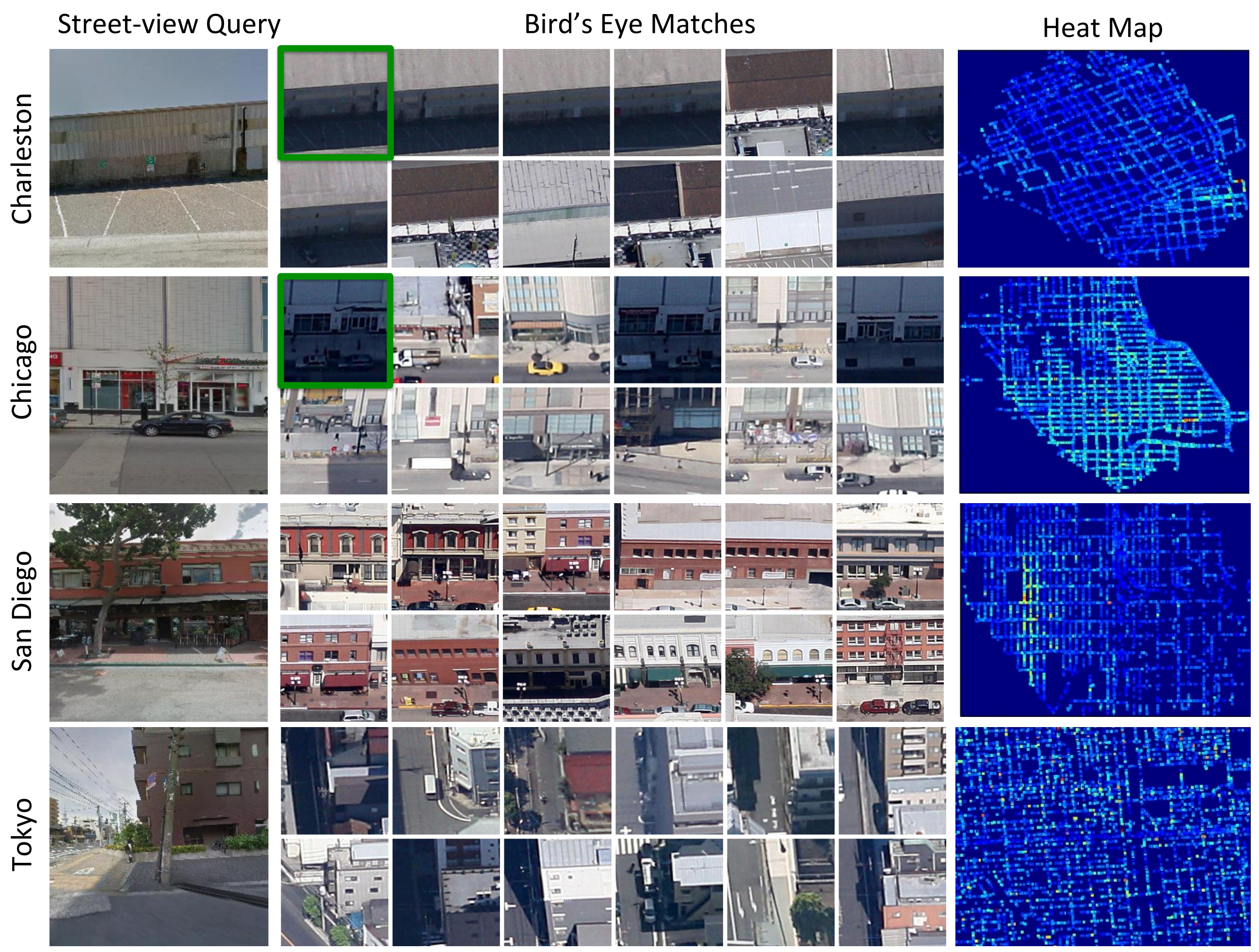


Figure 13: Geolocalization examples. For each query on the left, the top 12 matching aerial view crops are shown. The heat map on the right is colored as a function of the matching distance between the street-view query and the aerial crop at that location.

While the query image in the third and fourth row don't match the aerial views, the top retrievals look similar to the query image and the heat map activates in the city region that with most similar buildings. Note that for the query in Tokyo, it finds semantically similar buildings located at road intersections. The result suggests that Where-CNN-DS can generalize to unseen data reasonably well.

6. Conclusion and Discussion

We have presented the first general technique for the challenging problem of matching street-level and aerial view images and evaluated it for the task of image geolocalizaiton. While standard keypoint matching or bag-of-words approaches barely outperform chance, our learned representations show promise.

While we train and test on cross-view pairs that have been roughly aligned according to aerial and street-view metadata, a limitation of the current approach is the need to estimate scale and dominant depth at test time for ground-level queries with no metadata. This is plausible either through manual intervention or automatic estimation. Another limitation is that the absolute orientation of a ground-level query could be unknown (and difficult to estimate) and would require a sweep over orientations at test time.

Acknowledgments. Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory, contract FA8650-12-C-7212. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

References

[1] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: Capturing the world at street level. *Computer*, 2010. 2

[2] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *ECCV*, 2012. 2

[3] M. Bansal and K. Daniilidis. Geometric urban geo-localization. In *CVPR*, 2014. 2

[4] M. Bansal, K. Daniilidis, and H. S. Sawhney. Ultra-wide baseline facade matching for geo-localization. In *ECCV Workshops*, 2012. 2

[5] T. Berg and P. N. Belhumeur. Tom-vs-Pete classifiers and identity-preserving alignment for face verification. In *BMVC*, 2012. 1

[6] T. Berg and P. N. Belhumeur. POOF: Part-Based One-vs-One Features for fine-grained categorization, face verification, and attribute estimation. In *CVPR*, 2013. 1

[7] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002. 4

[8] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *CVPR*, 2013. 1

[9] D. M. Chen, G. Baatz, K. Köser, S. S. Tsai, R. Vedantham, T. Pylvänäinen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on mobile devices. In *CVPR*, 2011. 2

[10] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 1, 3

[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 5

[12] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *ICML*, 2006. 5

[13] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013. 5

[14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010. 5

[15] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *Proceedings of the international conference on very large data bases*, 1999. 4

[16] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 4

[17] J. Hays and A. A. Efros. im2gps: estimating geographic information from a single image. In *CVPR*, 2008. 2

[18] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, University of Massachusetts, Amherst, 2007. 1

[19] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. *h ttp://caffe. berkeleyvision. org*, 2013. 5

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 4, 5

[21] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*, 2010. 2

[22] T.-Y. Lin, S. Belongie, and J. Hays. Cross-view image geolocalization. In *CVPR*, 2013. 2, 7

[23] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 3

[24] Q. Shan, C. Wu, Y. F. Brian Curless, C. Hernandez, and S. M. Seitz. Accurate geo-registration by ground-to-aerial image matching. *3DV*, 2014. 2

[25] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 1, 3, 5

[26] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. In *JMLR*, 2008. 6

[27] J. Wang, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, Y. Wu, et al. Learning fine-grained image similarity with deep ranking. *arXiv preprint arXiv:1404.4661*, 2014. 3, 5

[28] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 5

[29] A. R. Zamir and M. Shah. Accurate image localization based on google maps street view. In *ECCV*, 2010. 2

[30] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 5