Adventures in Archiving and Using Three Years of Webcam Images

Anonymous IV submission

Paper ID 13

Abstract

Recent descriptions of algorithms applied to images archived from webcams tend to underplay the challenges in working with large data sets acquired from uncontrolled webcams in real environments. In building a database of images captured from 1000 webcams, every 30 minutes for the last 3 years, we observe that these cameras have a wide variety of failure modes. This paper details steps we have taken to make this dataset more easily useful to the research community, including (a) tools for finding stable temporal segments, and stabilizing images when the camera is nearly stable, (b) visualization tools to quickly summarize a years worth of image data from one camera and to give a set of exemplars that highlight anomlies within the scene, and (c) integration with LabelMe, allowing labels of static features in one image of a scene to propagate to the thousands of other images of that scene. We also present proof-of-concept algorithms showing how this data conditioning supports several problems in infering properties of the scene from image data.

1. Introduction

Cameras connected to the Internet, webcams, provide a large and continuous source of images of many locations around the world. Webcams tend to offer images of the same scene over a long period of time, offering the potential to learn more about a scene than is possible from, for example, a single image uploaded to Flickr. The Archive of Many Outdoor Scene (AMOS) data set [4] collects images from a large number of webcams offer an opportunity to study this problem domain. This data set has been used for studies in webcam geolocation, geo-orientation, and scene annotation, but often scenes are cherry picked to avoid problems caused by cameras which break, move (either drifting or suddenly as in Figure 1), or otherwise have problems.

In this paper, we report on progress in augmenting the AMOS database with side information to mitigate these problems and support additional inference tasks. This includes manually specifying, for each camea, temporal intervals when the images are static or nearly static, then automatically computing the warping parameters for exactly aligning each image within a segment. Also, we develop efficient tools to summarize the variability of a camera over the course of a year. Third, we integrate the LabelMe image labeling tool to allow features in the scenes to be labeled. This provides a compelling effort-multiplier effect, because the label of part of a scene within one image can be propagated to all images of that scene. This allows a quick way to mark parts of the image with non-visual information (such as watermarks and time stamps), and makes it easy to get, for example, thousands of pictures of the same tree under different lighting, seasonal and weather conditions.

Figure 1. Algorithms exist that can extract information from outdoor cameras and the scenes they view by making significant assumptions, such as known camera calibration or the absense of camera motion. Unfortunately these assumptions are often violated by real outdoor cameras. This paper presents our extensions to the ground truth labels available for a large dataset of images from webcams. We also present results that use the dataset for several novel applications.

We believe the effort to annotate, stabilize and visualize large webcam archives supports a large collection of interesting computer vision problems, especially in terms of long term inference over scene appearance and the natural causes of change that affect scene appearance. We conclude this paper with two proof-of-concept demonstrations. First is an example of image denoising showing dramatic results

in night-time images. Second, we offer an approach to take a set of images and a side channel of weather information (such as wind speed or vapor-pressure), create a linear predictor of the environmental variables directly from the image data. We offer these demonstrations not as complete or optimal solutions to their respective problems, but rather to highlight the potential and value of a well conditioned webcam imagery archive.

1.1. Related Work

Our work is related to many different areas of computer vision, here we describe work related to large dataset creation and algorithms designed to operate on webcam image sequences.

The creation of large labeled image datasets are challenging efforts and represent a significant contribution to the community. Recent examples include datasets of many small images [17], with labeled objects [13], and of labeled faces [3]. Each of these datasets fills a niche by providing a different types of labeled images. Most similar to the AMOS dataset [4] is one with many images, and associated metadata, from a single carefully-controlled static camera [10]. The AMOS dataset is unique in providing time-stamped images from many cameras around the world. No other dataset provides the broad range of geographic locations and the long temporal duration. This paper presents new annotations that further increase the value of this large dataset.

Many algorithms have been developed to infer scene and camera information using long sequences of images from a fixed view. Examples include a methods for clustering pixels based on the surface orientation [8], for factoring a scene into components based on illumination properties [14], for obtaining the camera orientation and location [6, 5, 15, 9], and for automatically estimating the time-varying camera response function [7]. We present new results on estimating meterological properties from long sequences of images from a webcam.

Given the vast number of images in the AMOS dataset, nearly 40 million as of March 2009, it is often challenging to find the subset of images that are suitable for a particular algorithm evaluation. Compact summaries can enable rapid browsing of a large collection of images. One area of previous work is on image-based summaries of a video, see [11] for a survey. Another interesting approach uses a short video clip to summarize the activity in a much longer video [12]. To our knowledge, all the previous work is designed to work with high frame-rate video. We present visualizations that highlight the geographic nature and long-temporal duration while simultaneously handling a very low-frame rate and very long duration dataset.

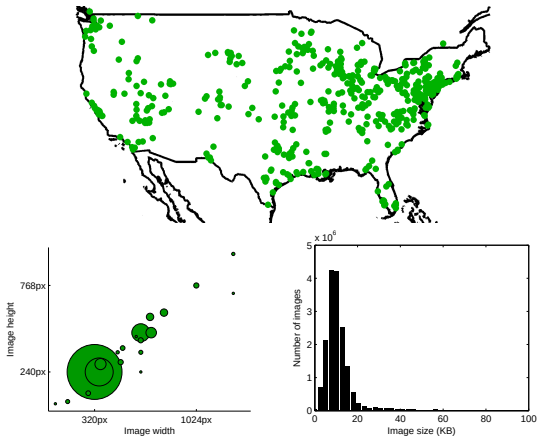


Figure 2. (top) A scatter plot of the locations of cameras in the AMOS dataset. (bottom-left) The distribution of image sizes measured in pixels. Each circle is centered at an image size with area proportional to the number of cameras that generate images of that size. (bottom-right) The distribution of image sizes in kilobytes.

2. AMOS: Archive of Many Outdoor Scenes

The AMOS dataset [4] consists of over 40 million images captured since March 2006 from 835 outdoor webcams located primarily in the continental United States (see Figure 2). This dataset is unique in that it contains significantly more scenes than in previous datasets [10] of natural images from static cameras.

The cameras in the dataset were selected by a group of graduate and undergraduate students using a standard web search engine. Many cameras are part of the Weatherbug camera network [18]. Images from each camera are captured several times per hour using a custom web crawler that ignores duplicate images and records the capture time. The images from all cameras are 24-bit JPEG files that vary in size from 49×46 to 3400×1600, with the majority being 320×240. The file size at the 1st, 50th, and 99th percentile are respectively 3kB, 11kB, and 79kB with a mean of 14 kB. See Figure 2 for more information about the distribution of image sizes and dimensions. The small image size is typical of webcam networks, and indicates the dramatic compression of each image; this motivates the problem domain of image denoising considered in Section 5.1.

In addition to a large amount of image data, each camera is assigned latitude and longitude coordinates; in most cases the coordinates are assigned by a human but in some cases the coordinates were estimated based on the camera IP address.

3. New Data Set Annotations

The value of the AMOS dataset is somewhat reduced by the high percentage of unsuitable images and cameras. Webcams can fail to generate "good" images for many differ-

ent reasons; the AMOS dataset contains examples of many common and uncommon failure modes. The following two sections describe annotations and visualizations that we are adding to help find subsets of images that are suitable for different potential applications.

3.1. Static cameras "in the wild"

Camera motion is a significant problem for algorithms that use long sequences of images of outdoor scenes. Long-term camera stability is a requirement of the algorithms proposed for applications such as camera geo-location [6], geo-orientation [5, 9], color analysis [15], and radiometric analysis [7]. We argue, with empirical justification, that truly static cameras rarely exist "in the wild".

We are manually labeling all the images in the AMOS data for the first three years. The labels consist of temporal intervals with one of the following labels: *static* for cameras with motion of one-pixel or less, *semi-static* for cameras with motion of less than 5 pixels (often due to wind, temperature changes, camera drift), and *non-static* for all other cases. We perform this labeling for each camera by inspection of all images captured at noon. This sparse set of labels is extended to all images by assuming that if two consecutive noon images from a single camera were *static* (*semi-static*) then the intervening non-noon images are also *static* (*semi-static*). Any interval that is not manually labeled as *static* or *semi-static* is considered *non-static*.

Results on the first 85 cameras of the AMOS dataset show that 21% of the images were labeled as *static*, 28% were labeled as *semi-static*. More than half of the images were in temporal intervals which include problems such as: optical or electrical corruption, multiplexing of images from many cameras, or continuous camera motion. Only two cameras were labeled as *static* for the full three years. While this labeling will be valuable for algorithm developers it also motivates the important problem of converting *semi-static* intervals to *static* intervals, in other words, solving for the camera motion and aligning the images. Section 3.2 describes a simple alignment algorithm and shows the impact of a successful alignment.

3.2. Automatic scene alignment

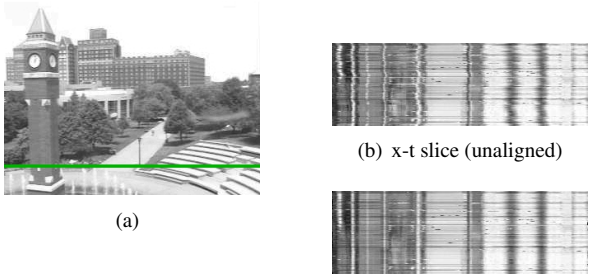
Alignment of the *semi-static* intervals would more than double the number of images in the AMOS dataset available to algorithms that require a *static* camera. There is a long history of work [16] on joint alignment of image sets and many of these techniques would work well for alignment of a small set of webcam images. This section describes a simple algorithm for scene alignment and highlights the benefits of scene alignment.

Our scene alignment algorithm uses the gradient magnitude image to reduce sensitivity to weather and illumination conditions. The algorithm initializes each image transformation to the identity and iterates over the following steps until convergence:

1. compute the average gradient magnitude image of the transformed images,
2. align each gradient magnitude image to the average gradient magnitude image using the Lucas-Kanade method with an affine motion model.

This simple scene alignment algorithm could be improved in many ways: using better image features for alignment, using a more robust image distance measure (such as mutual information), automatic support for large camera motions, and coarse-to-fine alignment. Developing a scene alignment technique capable of aligning all the images in the AMOS dataset with reasonable computational requirements is an important area for future work.

We qualitatively evaluated the algorithm on several *semi-static* intervals from the AMOS dataset. Figure 3 shows results of the scene alignment algorithm on one scene. To better understand the impact of this subtle realignment we compared the principal components of the aligned and unaligned sequences. The significant differences in both the components and the coefficients between the aligned and unaligned sequences are shown in Figure 4). For this scene, in the unaligned sequence the 3rd PCA component codes seems to code exclusively for camera motion. Removing this effect directly impacts PCA based algorithms for geo-location [6], and may make algorithms that infer lighting variation by looking at single pixel locations over time robust enough to not require hand chosen points.

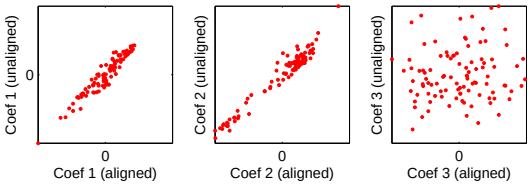


(a) (b) x-t slice (unaligned) (c) x-t slice (aligned)

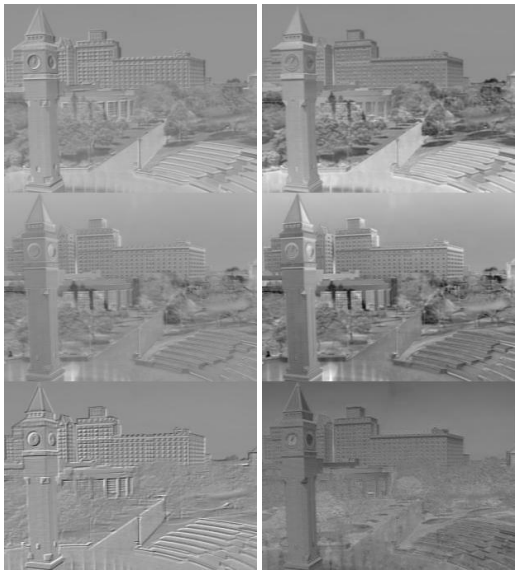
Figure 3. Results of the alignment procedure. (a) An image from the scene with a horizontal line that shows the image location of the x-t slices shown below. (b) An x-t slice from the unaligned image sequence. (c) An x-t slice from the aligned image sequence.

3.3. Object Labeling

Localized object annotations in images are valuable for learning-based methods and algorithm evaluation. The challenge is that it can be time consuming to label images. We use the stable segment annotations described in Section 3.1 to significantly reduce the cost of annotating objects



(a) PCA coefficient scatter plots



(b) PCA components

Figure 4. Removing jitter from a sequence of images can significantly change the principal components of the sequence. (a) Scatter plots of the top three PCA coefficients for the aligned and unaligned sequences. The top two coefficients are highly correlated. The impact of the alignment step is clearly evident in the third coefficient (right). (b) The top three principal component images for the unaligned (left) and aligned (right) sequence. The third component of the unaligned sequence (bottom-left) is significantly different from that of the aligned sequence.

in the AMOS dataset. We make the observation that object annotations from a single frame can be extended through time if the frame is during a stable segment. We have integrated the LabelMe annotation tool [13] into the AMOS website. Using this tool it will be possible to annotate static scene elements and obtain views of the same scene element in many weather conditions and seasons. Figure 5 shows an example of one such annotation extended through time. Another use for this tool is to label potentially non-interesting image features such as watermarks and time-stamps.

4. Visualization Tools

Building a system that works with openly available cameras on the web requires an acceptance that cameras may not be consistently available. Finding times when a camera has moved is one vital step in preparing long term time-

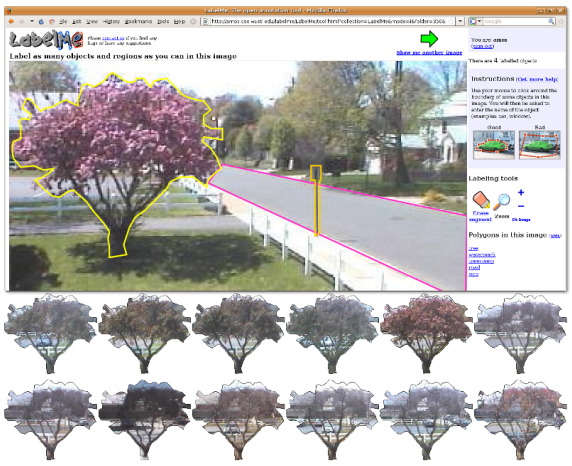


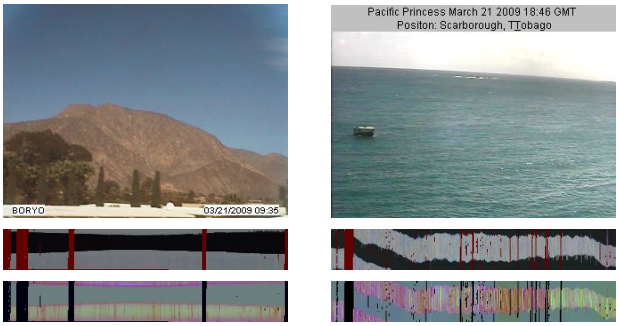
Figure 5. We use the LabelMe annotation tool [13] to rapidly annotate many images from a webcam. The annotations from a single image extended through time during periods without camera motion.

lapses for further use, but there are other scene changes that may impact further analysis. This section describes efficient tools for creating natural visualizations to summarize the overall appearance changes over the course of years. We also present tools to quickly find a representative set of the most anomalous images.

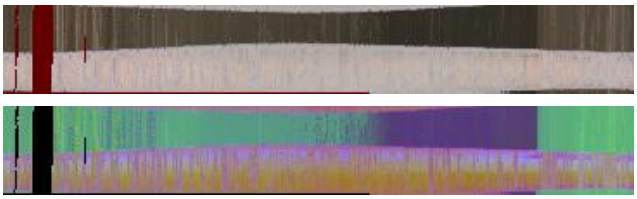
4.1. Summary Visualizations

Given a set of images $I_1 \dots I_n$ captured with time and date stamps, creating annual summary images requires simply converting the irregular sampling of the time and date into a regular sampling suitable for displaying as an image. We compute the mean RGB values for each image, and specify a desired regular sampling of the dates and time (365 days per year, and 48 times per day), and use nearest neighbor interpolation to find the image sample closest to each desired sample. If the distance to the nearest neighbor is greater then or equal to one unit of the regular sampling (i.e., a day in the date dimension, or half and hour in the time dimension) then we color that pixel dark red. Dark red was prefered over white or black or gray because no cameras in our collection ever have a mean image value of dark red. Figure 6 shows two example cameras and the annual summaries of those cameras for 2008.

This summary image serves to show gross patterns of image availability, but there may be important image changes which do not affect the mean image intensity. The following is a similarly concise summary visualization tool which highlights more of the changes within the scene. Each image I_j, $j \in [1 \dots n]$ is written as a column vector, and a data matrix $\mathbf{I} \in \mathbb{R}^{p \times n}$ is constructed by concatenating these column vectors. This data matrix is quite large (as we have about 17000 images per cameras), so we use an incremen-



(a) Two cameras and their summary visualizations



(b) 2008 Summary of camera in Figure 4

Figure 6. The top shows images from two cameras in our database, and the RGB then the PCA based annual summary image. The right camera is on the bridge of a Princess Cruise line ship, which circumnavigated the globe during 2008, explaining the fact that when night occurs "wraps" during the year. At the bottom is the visualization of the camera shown in Figure 4. Here is an example where the RGB shows clearly the day-night distinction, but the PCA color coded also highlights the fact that the images appear very different in the morning vs. the evening (the light blue to yellow variation near the bottom edge), and the fact that the night image has different appearances, caused by a bright light which is on at some parts of the year and not others.

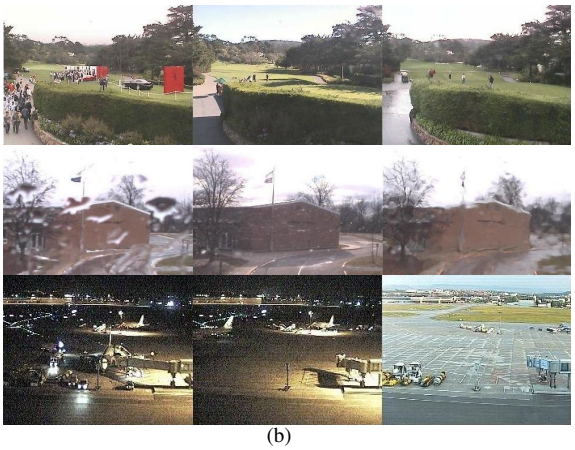
tal SVD algorithm [1] to perform the PCA decomposition to recover the decomposition $\mathbf{I} \approx USV$. We compute the top three components in the PCA decomposition, so U is a $p \times 3$ matrix of component images and V is a $3 \times n$ matrix of coefficients. We create our false color annual summary visualization based on the coefficients V. These are linearly normalized to vary between 0 and 1, and used as the "mean RGB values" for each image, and the same nearest neighbor interpolation is used as before (except that missing values can now be safely coded as black). The bottom of each part of Figure 6 shows this visualization for 3 different cameras.

4.2. Anomalous Images

Finding and showing exemplar images is a useful visualization tool for summarizing long image sequences [11]. This section describes a simple method that selects the most anomalous images using PCA and describes a computationally-efficient improvement that reduces the redundancy of the set of exemplars.

We first compute a PCA basis of a set of images $\mathbf{I}$ from a single camera and assign a score to each image that reflects

(a)



(b)

Figure 7. The most anomalous images often provide an interesting overview of the activity in the scene. (a) The three most anomalous images in a scene selected using the naïve method. This method often selects redundant images. (b) Examples, from several scenes, of the most anomalous images generated using the method described in Section 4.2. Note that for the first example the reduncant image with the red flags is not shown.

.

the degree to which the image is anomalous (we use a ten dimensional PCA basis for this section). The score we use is the SSD reconstruction error with respect to the basis. The exemplars are simply the images with the largest reconstruction error, we choose the top three. Figure 7 shows exemplars obtained using this method for one scene. As shown in the figure, for some webcams many of the most anomalous images are similar to each other. Showing many examples of very similar images may not provide a useful overview of the anomalous images of a scene.

To address this problem we propose the following method to select exemplars. We select the top n most anomalous images $\hat{\mathbf{I}} = \{I_1, ..., I_n\}$ as candidate exemplars. Then we begin by including the most anomalous image in the set of exemplars. For each subsequent exemplar we choose the image from $\hat{\mathbf{I}}$ that is furthest, in Euclidean distance, from any image already in the exemplar set. Results using this improvement (see Figure 7) show that the method reduces the problem of finding redundant exemplars.

5. Applications

In this section we offer two proof-of-concept demonstrations of potential applications made possible by analysis of long image sequences from a fixed camera. The first appli-

cation is in image denoising, taking advantage of the stable camera to efficiently characterize the distributions of local image appearance; the second explores the relationship between images and meteorological quantities such as wind speed and vapor pressure.

5.1. Image Denoising

Because webcams have limited ability to adjust to light conditions (for example they have small maximum apertures), and because they are often highly compressed (to reduce bandwidth), they are often extremely noisy. One recently popularized method of image denoising is based on non-local averaging [2], from which the following description is based.

Given a discrete noisy image, where $I(i)$ defines the intensity at pixel i, the non-local mean image $I_{NL}(i)$ is computed for a pixel i as a weighted average of all the pixels in the image,

$$I_{NL}(i) = \sum_{j \in I} w(i,j) I(j), \tag{1}$$

subject to the conditions that $0 \leq w(i,j)$ and $\sum_j w(i,j) = 1$. Unlike common image blurring algorithms, where $w(i,j)$ depends on the distance between the pixel locations, in non-local averaging, the weight function depends on the difference between the local neighborhood around pixels i and j,

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{||I(\mathcal{N}_i) - I(\mathcal{N}_j)||}{\sigma^2}}, \tag{2}$$

where Z(i) is the normalizing constant and $\mathcal{N}_i$ is the local neighborhood of pixel i.

Non-local averaging gives interesting denoising results because natural images contain redundant structure. An image patch, for example, along the edge of a roof is likely to be similar to other patches along the same side of the same roof, so averaging similar patches gives a noise reduction without blurring the scene structure.

This is even more sure to be true in the case of static images captured over a long time period, because one patch views *exactly* the same scene elements over time. Thus non-local temporal averaging uses exactly the same formulation, except that the non-local temporal average is computed as the weighted sum of the *same* pixel (in images taken at different times). Extending the notation above to define $I(i,t)$ as the intensity of pixel i during frame t, we specify our non-local temporal average image I_{NLT} as:

$$I_{NLT}(i,t) = \sum_{t' \in T} w(i,t,t') I(i,t') \tag{3}$$

Figure 8 shows one example of this image denoising, applied to debluring night-time image of gate at an airport using a set of images captured once per day (at the same time each day), in a scene in which the airplane is often missing

Figure 8. A noisy webcam image, and a version with noise reduced using non-local (temporal) averaging.

and never in the same place and the jet-bridge often moves. Although this result is anecdotal, simple averaging would clearly fail, and the non-local temporal result shows substantial noise reduction without any blurring of the features in the scene.

5.2. Using images as environmental sensors

Local environmental properties often directly affect the images we collect from the webcams; whether it is cloudy or sunny is visible by the presence of shadows; wind speed and direction is visible in smoke, flags, or close up views of trees; particulate density is reflected in haziness and the color spectrum during sunset. We explore techniques to automatically extract such environmental properties from long sequence of webcam images. This allows the webcams *already* installed across the earth to act as generic sensors to improve our understanding of local weather patterns and variations.

We consider two weather signals for our driving examples: wind velocity and vapor pressure. These two signals present unique challenges and opportunities. The effect of wind velocity is limited to locations in the scene that are affected by wind (flags and vegetation) while the effect of vapor pressure on the scene may result in broad, but subtle, changes to the image. Our method assumes the availability of images and weather data with corresponding timestamps. Further, given the localized nature of much weather data it is most useful if the weather data is collected near the camera.

The first step of our method is to extract significant scene variations using PCA with 10 components ($k = 10$). The coefficients used to reconstruct each image define low-dimensional time-stamped summaries $V \in \mathbb{R}^{k \times n}$ of the scene variation. In the second step, Canonical Correlation Analysis (CCA) to relate the time-series of a weather signal with the corresponding coefficients V of the images from the camera. Unlike PCA which finds projections that maximize the covariance of a single dataset, CCA finds a pair of projections that jointly maximize the covariance of the pair of datasets. In this case, given image principal coefficients V and weather data $Y \in R^{m,n}$, CCA finds two matrices A, B such that $AV \approx BY$. The matrices A and B enable

prediction of weather data from an image and vice versa. To predict the weather data given a new image I_i we first project it onto the PCA basis to obtain the principal coefficients v_i. Using these coefficients we predict the value of the weather signal y_i as $y_i = Av_iB^{-1}$.

We now consider two examples to evaluate our method. As input we use images from the AMOS dataset and weather data from the Historical Weather Data Archives (HDWA) maintained by the National Oceanic and Atmospheric Administration (NOAA). We use the ground truth location of the camera to find the location of the nearest weather station and use the provided web interface to download the desired data. In both cases we solve for PCA and CCA projections using two hundred images captured during midday for approximately two months and evaluate on one hundred images from the following several weeks.

The first example is in predicting wind velocity. We find that CCA computes a pair of matrices A, B that approximate a linear projection of the wind velocity. Figure 9 shows results including the linear image projection found by our method. This projection (the canonical correlations analogy to a principle image component) is plausible and clearly highlights the orientation of the flag. The plot shows the first dimension that CCA predicts from both the webcam images and the weather data for our test data. The prediction of the second dimension (not shown) is much less accurate which means that for this scene our method is able to predict only one of two components of the wind velocity. This result is not surprising because the image of the flag in the scene would be nearly identical if the wind was blowing towards or away from the camera. In Figure 10 we show the relationship of the CCA projection vector and the geographic structure of the scene. We find that the wind velocity projection vector is, as one would expect, perpindicular to the viewing direction of the camera.

As a second example we use a different scene and attempt to predict the vapor pressure, the contribution of water vapor to the total atmospheric pressure (we note that since vapor pressure is a scalar the CCA based method is equivelent to linear regression). Using the method exactly as described for predicting wind velocity in the previous example fails, in other words no linear projection of a webcam image is capable of predicting vapor pressure. We find that replacing the original images with the corresponding gradient magnitude image achieves much better results. Figure 11 shows vapor pressure prediction results using the gradient magnitude images. The results show that vapor pressure is strongly related to differences in gradient magnitudes in the scene.

6. Conclusion

Collecting and sharing a large dataset of images is a challenging and time consuming task. This is especially true for the AMOS dataset due to the numerous common and uncommon camera failure modes. In this paper, we described additional annotations being added to the dataset. In addition, we presented several visualization that make it easier to find suitable image subsets.

The AMOS dataset make possible empirical evaluations that were once untenable. We believe that inference algorithms are possible to predict many meteorological and environmental properties directly from image data, and our proof-of-concept demonstrations in estimating wind speed and vapor pressure suggest that this is a viable direction of future work. This motivates our efforts to continue to collect, organize, analyse, and distribute this dataset for the computer vision community.

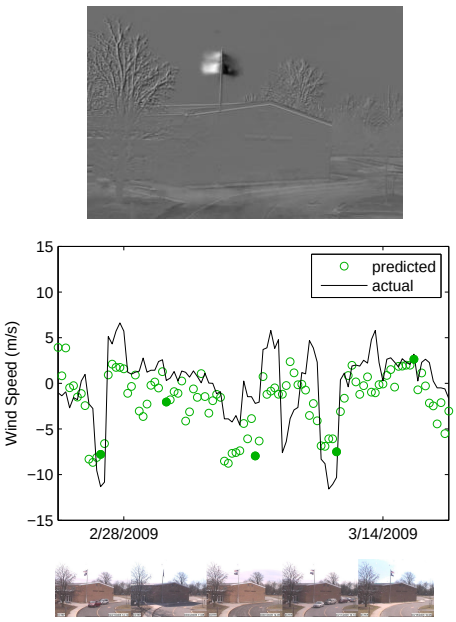


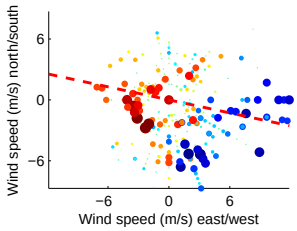
Figure 9. An example of predicting wind speed in meters per second from webcam images. (top) The projection of the gradient magnitude used to linearlly predict the wind speed. (middle) Predicted wind speed values and corresponding ground truth. (bottom) Each image corresponds to a filled marker in the plot above. See Figure 10 for futher verification of these predictions.

References

[1] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Proc. European Conference on Computer Vision*, pages 707–720, 2002.

[2] A. Buades, B. Coll, and J.-M. Morel. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 60–65, 2005.

[3] 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 CS TR-07-49, University of Massachusetts, Amherst, 2007.



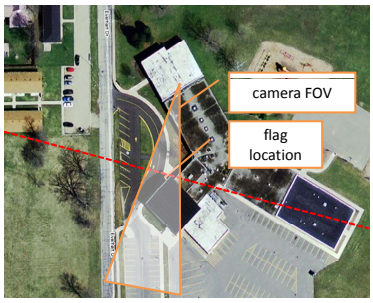


Figure 10. Further analysis of the wind velocity prediction experiment in Figure 9. (top) A scatter plot of wind velocities. The position of each marker is determined by the wind velocity measured by a weather station. The color and size of each marker is determined by wind speed predicted from a webcam image archived at the same time as the wind measurement. The dashed line (red) is the projection axis, determined using CCA, used to predict wind speed (the actual wind speed values shown in Figure 9) from wind velocity. (bottom) An image from Google Maps of the area surrounding the camera. The camera FOV was crudely estimated by visually aligning scene elements with the satellite view. The dashed line (red) is equivelent as the dashed line (red) in the plot above. This image confirms that, as one would expect, our method is best able to predict wind direction when the wind approximately perpendicular to the principal axis of the camera.

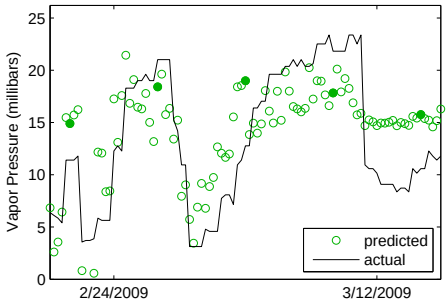


Figure 11. An example of predicting vapor pressure (a meteorological quantity) from webcam images. (top) The projection of the gradient magnitude used to linearlly predict the vapor pressure. (middle) Predicted vapor pressure values and corresponding ground truth. (bottom) Each image corresponds to a filled marker in the plot above. Inspection of the images revealed that the poor prediction accuracy for the period following March 12, 2009 was due to heavy fog and subsequent water on the optics.

[4] N. Jacobs, N. Roman, and R. Pless. Consistent temporal variations in many outdoor scenes. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, June 2007.

[5] N. Jacobs, N. Roman, and R. Pless. Toward fully automatic geo-location and geo-orientation of static outdoor cameras. In *Proc. IEEE Workshop on Applications of Computer Vision (WACV)*, Jan. 2008.

[6] N. Jacobs, S. Satkin, N. Roman, R. Speyer, and R. Pless. Geolocating static cameras. In *Proc. IEEE International Conference on Computer Vision*, Oct. 2007.

[7] S. J. Kim, J.-M. Frahm, and M. Pollefeys. Radiometric calibration with illumination change for outdoor scene analysis. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[8] S. J. Koppal and S. G. Narasimhan. Clustering appearance for scene analysis. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1323–1330, 2006.

[9] J.-F. Lalonde, S. G. Narasimhan, and A. A. Efros. What does the sky tell us about the camera? In *Proc. European Conference on Computer Vision*, 2008.

[10] S. G. Narasimhan, C. Wang, and S. K. Nayar. All the images of an outdoor scene. In *Proc. European Conference on Computer Vision*, pages 148–162, 2002.

[11] J. Oh, Q. Wen, J. Lee, and S. Hwang. Video abstraction. *Video Data Management and Information Retrieval*, pages 321–346, 2004.

[12] Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg. Webcam synopsis: Peeking around the world. pages 1–8, Oct. 2007.

[13] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, May 2008.

[14] K. Sunkavalli, W. Matusik, H. Pfister, and S. Rusinkiewicz. Factored time-lapse video. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3), Aug. 2007.

[15] K. Sunkavalli, F. Romeiro, W. Matusik, T. Zickler, and H. Pfister. What do color changes reveal about an outdoor scene? *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[16] R. Szeliski. Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis.*, 2(1):1–104, 2006.

[17] A. Torralba, R. Fergus, and W. T. Freeman. Tiny images. Technical Report MIT-CSAIL-TR-2007-024, Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, 2007.

[18] Weatherbug Inc. http://weatherbugmedia.com/.