Industrial Smoke Detection and Visualization

Yen-Chia Hsu, Paul Dille, Randy Sargent, Illah Nourbakhsh

CMU-RI-TR-16-55

September 2016

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Keywords: Data visualization, computer vision, air quality, community engagement

Acknowledgments

This work is supported by the Heinz Endowments. The authors thank the Allegheny County Clean Air Now (ACCAN) community for participating in using the system presented in this work. The authors also thank Aaron Steinfeld and Srinivasa Narasimhan in CMU Robotics for providing academic feedback.

Contents

Abstract

As sensing technology proliferates and becomes affordable to the general public, there is a growing trend in citizen science where scientists and volunteers form a strong partnership in conducting scientific research including problem finding, data collection, analysis, visualization, and storytelling. Providing easy-to-use computational tools to support citizen science has become an important issue. To raise the public awareness of environmental science and improve the air quality in local areas, we are currently collaborating with a local community in monitoring and documenting fugitive emissions from a coke refinery. We have helped the community members build a live camera system which captures and visualizes high resolution timelapse imagery starting from November 2014. However, searching and documenting smoke emissions manually from all video frames requires manpower and takes an impractical investment of time. This paper describes a software tool which integrates four features: (1) an algorithm based on change detection and texture segmentation for identifying smoke emissions; (2) an interactive timeline visualization providing indicators for seeking to interesting events; (3) an autonomous fast-forwarding mode for skipping uninteresting timelapse frames; and (4) a collection of animated smoke images generated automatically according to the algorithm for documentation, presentation, storytelling, and sharing. With the help of this tool, citizen scientists can now focus on the content of the story instead of time-consuming and laborious works.

Chapter 1

Introduction

As sensing devices become available and affordable to the general public, there is a growing trend in citizen science [10] where non-professionals find and define problems, collect and analyze environmental data by using sensors in order to better understand their surroundings, and to tell compelling stories in communicating findings. Storytelling is an important technique in communication, since visual stories framed by data are easy to remember, encourage discussions, support decision making, and have the potential to raise public awareness [13, 16]. However, citizen scientists often lack related skills and require the assistance of experts in building sensor networks, analyzing large-scale environmental data, and integrating the findings into scientific stories. Therefore, developing and providing easy-to-use tools matching the needs of local communities is vital in helping them improve technology fluency [21].

We are currently collaborating with a local community in Pittsburgh in gathering air quality data (particle pollution PM2.5), documenting images of fugitive emissions from a coke refinery, and presenting scientific stories about how smoke emissions affect the local air quality. Figure 1.1 demonstrates smoke emissions with various lightings, appearance, and opacities. Figure 1.2 shows steam, shadow, and the mixture of steam and smoke.

We have helped the local community build a live camera monitoring system. The system collects high resolution imagery starting from November 2014 and visualizes the results by using a web-based large-scale timelapse viewer which we developed previously [1, 19]. The local community can utilize the viewer to explore the high quality time-series images by panning and zooming, finding fugitive emissions, and using the provided thumbnail tool to generate and share animated images. Nevertheless, the speed of receiving images greatly exceeds the time required to process them. The live camera takes a picture every 5 seconds and a timelapse for one day contains nearly 17,000 frames. Manually searching through each image to identify smoke emissions takes more than six times as long without computer vision automation. This paper presents a computer vision tool for detecting industrial smoke emissions and generating related animated images automatically, which significantly reduces the workload of citizen scientists. The task is to detect frames from a static camera containing smoke, exclude the ones having steam and shadow, identify the starting and ending frames of emissions, and output animated images which may include smoke. The local community can easily select representative images and insert them directly into Google Docs as animated image sequences to form a collection of fugitive emissions.

Figure 1.1: This figure shows emissions with various lightings, appearance, and opacities.

Figure 1.2: This figure shows steam, shadow, and the mixture of steam and smoke.

Chapter 2

Related Work

There are three general approaches appearing in previous research for detecting the presence of smoke emissions in a single image or across multiple frames: (1) color modeling; (2) change detection; and (3) texture analysis.

Color modeling describes the characteristics of image intensity values. For instance, smoke is grayish and has low saturation. Previous research used color models to identify smoke pixels [4] or extract color histogram features [15].

Change detection [18] determines moving objects in an image, which provides candidate regions containing smoke emissions for further analysis. One common technique is background subtraction [5, 6] which estimates an image without moving objects from an image sequence, subtracts the estimated image from the current one to get a residual image, and thresholds the residual image to obtain a binary mask. In addition, there are background modeling approaches [8, 22] which learn a probabilistic model of each pixel using a mixture-of-Gaussians and determine the background pixels according to the probability distribution. Other techniques involve computing the entropy of the optical flow field [12] to identify smoke and checking flickering pixels at the edge of candidate smoke regions [24].

Texture analysis measures texture energy in a single image or texture changes between multiple frames. One approach is to apply texture descriptors, such as a wavelet transform, on small blocks in an image for obtaining feature vectors and train a classifier using these features [3, 9].

Each of these approaches has distinct strengths and weaknesses. Color modeling is straightforward, but suffers from situations where smoke and non-smoke objects have the same chrominance (e.g. white smoke and steam, dark shadow and black smoke) or the background does not contain plentiful color information due to various weather and lighting conditions (e.g. fog, nighttime images). Background subtraction and background modeling do not distinguish smoke from non-smoke regions since they find all moving objects including shadow, steam, and smoke. Optical flow can determine smoke motions, but has high computational cost. It is difficult to extract useful information from texture analysis if the background does not contain sufficient texture information. Several research has integrated these methods into a system for better performance. Toreyin et al. [24] combined background subtraction, edge flickering, and texture analysis into a final result. Lee et al. [15] used change detection to extract candidate regions, computed feature vectors based on color modeling and texture analysis, and trained a support vector machine classifier using these features.

We are aware of other advanced machine learning approaches. For instance, Hohberg [11] trains a convolutional neural network for recognizing wildfire smoke. Tian et al. [23] present a physical based model and use sparse coding to extract reliable features for single image smoke detection. However, a simpler heuristic approach combining color modeling, change detection, and texture analysis is sufficient for our current needs.

Chapter 3

System

This chapter describes the method of the smoke detection algorithm, the experiment for evaluating the performance, and the visualization for showing smoke detection results.

3.1 Method

Inspired by prior method integration approaches, we have implemented a smoke detection algorithm for detecting fugitive emissions during the daytime from a static camera. The algorithm contains five steps: preprocessing, change detection, texture segmentation, region filtering, and event detection. Change detection identifies moving pixels containing smoke, steam, and shadow. Texture segmentation clusters pixels into several candidate regions based on texture information. Region filtering iteratively evaluates each candidate region based on shape, color, size, and the amount of change to determine if it matches the appearance and behavior of smoke. Event Detection groups video frames with smoke together to identify the starting and ending time of fugitive emissions.

3.1.1 Preprocessing

We apply the algorithm on 9700 daytime frames for each day and ignore nighttime. To reduce the computational cost, we first scale the original image at time t down to one-fourth of the original size to obtain a downsampled image I_t. Then we estimate the background image B_t by taking the median over the previous 60 images as shown in (3.1).

$$B_t(x, y) = \text{median}\big(I_t(x, y), ..., I_{t-59}(x, y)\big) \tag{3.1}$$

where (x, y) indicates the position of a pixel. Finally we convert all RGB images with 8-bit unsigned integer format to double precision ranging from 0 to 1.

3.1.2 Change Detection

Change detection finds moving pixels in video frames by computing changes in high frequency signals (e.g. edges, textures) and image intensity values (e.g. colors).

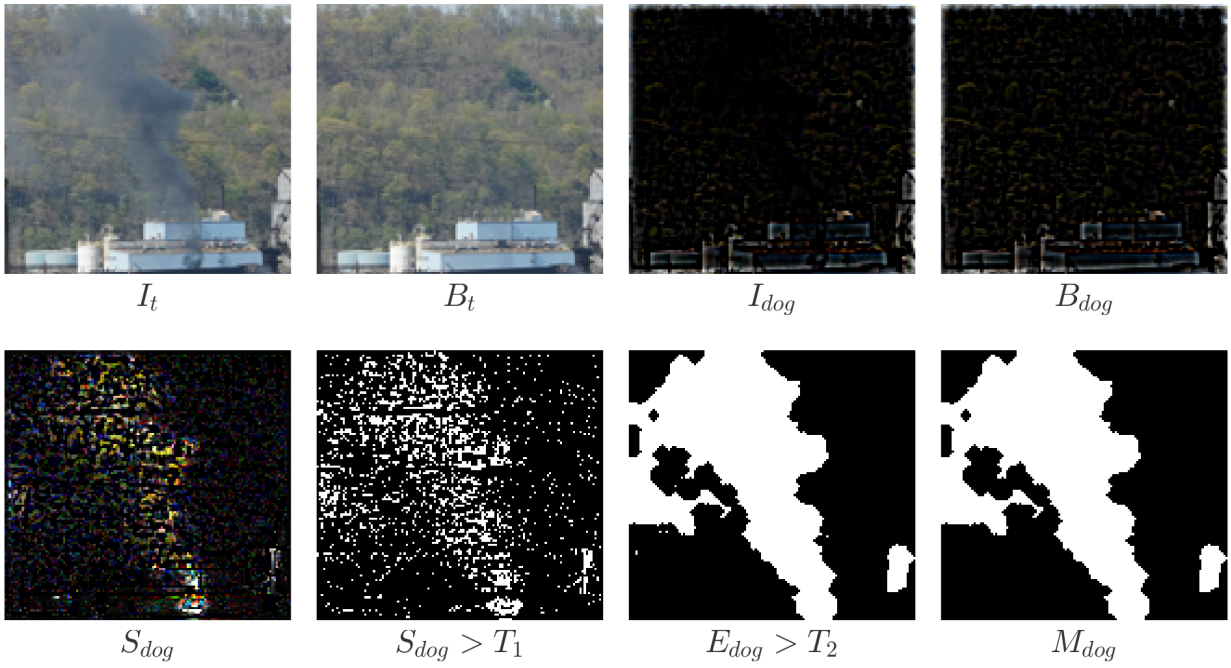

Figure 3.1: This figure visualizes the steps of high frequency change detection. Refer to section 3.1.2 for detailed explanation.

High Frequency Change Detection

Smoke is semi-transparent with various opacities and occludes parts of the background upon presence, which causes changes of high frequency signals across frames. First we compute the difference of Gaussian (DoG) of I_t and B_t to obtain I_{dog} and B_{dog} as shown in (3.3)

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\Big(-\frac{x^2+y^2}{2\sigma^2}\Big) \tag{3.2}$$

$$\begin{aligned} I_{dog}(x,y) &= \big(G_{\sigma_1}(x,y) - G_{\sigma_2}(x,y)\big) * I_t(x,y) \\ B_{dog}(x,y) &= \big(G_{\sigma_1}(x,y) - G_{\sigma_2}(x,y)\big) * B_t(x,y) \end{aligned} \tag{3.3}$$

where the asterisk sign $*$ indicates the convolution operator and $G_\sigma(x,y)$ is a Gaussian kernel with variance σ^2 and mean zero. The DoG image contains high frequency information for the current and the background images.

Then we perform background subtraction on I_{dog} and B_{dog} to obtain $S_{dog} = \mathrm{bgSub}(I_{dog}, B_{dog})$ as shown in (3.4).

$$\mathrm{bgSub}(I,B) = \frac{|I-B|}{\max\big(I+B, 0.1\big)} \tag{3.4}$$

Dividing the background subtraction term in the nominator by $\max\big(I+B, 0.1\big)$ alleviates the effect of illumination in images. The $\max$ function in the denominator in (3.4) prevents dividing to an extremely small value or zero. One way to interpret the S_{dog} image is that it measures the change of high frequency signals such as edges and texture between the current and background

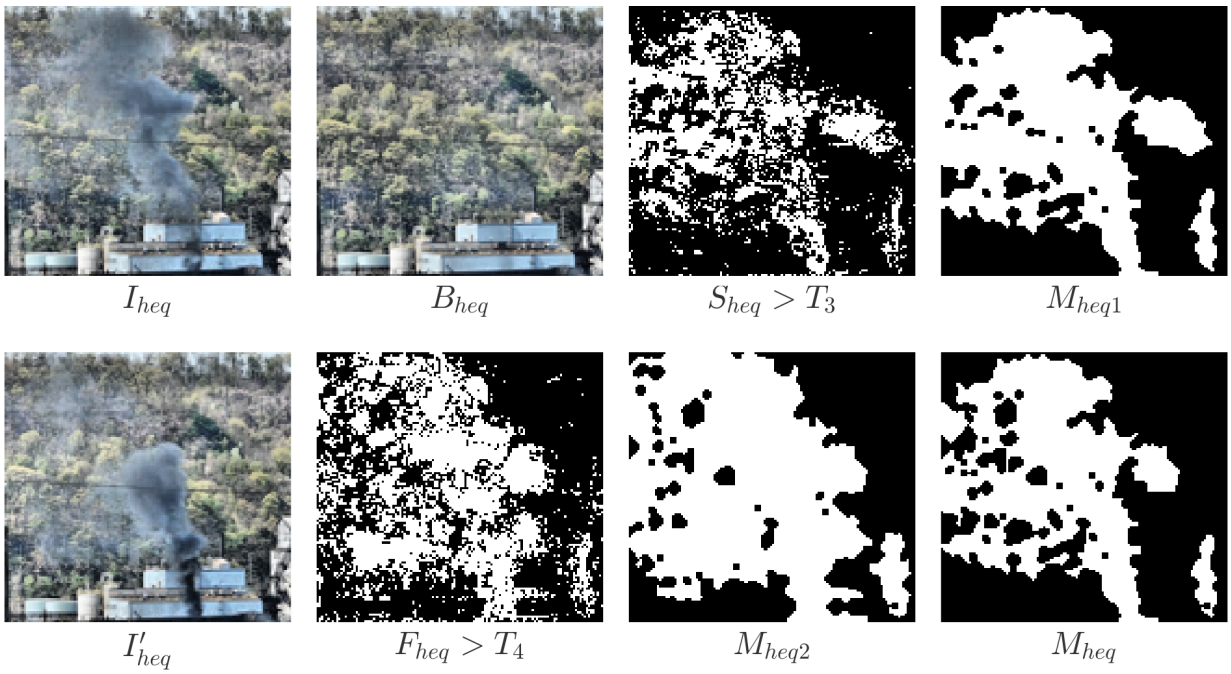

Figure 3.2: This figure visualizes the steps of image intensity change detection. Refer to section 3.1.2 for detailed explanation.

image. Thresholding channels in S_{dog} yields a binary image. Computing the local entropy of the 9-by-9 neighborhood centered around each pixel in the binary image gives an entropy image E_{dog} as show in (3.5).

$$E_{dog} = \text{entropyFilter}\big(\text{bgSub}(I_{dog}, B_{dog}) > T_1\big) \tag{3.5}$$

Finally we threshold the entropy image E_{dog} to obtain a binary image $E_{dog} > T_2$. Performing morphological closing, removing noise using a median filter, and discarding small regions using connected component algorithm on the binary image yields the smoothed image M_{dog} as shown in (3.6).

$$M_{dog} = \text{smooth}(E_{dog} > T_2) \tag{3.6}$$

Figure 3.1 visualizes the steps of high frequency change detection. If the M_{dog} image contains no regions (i.e. all pixel values are zero), the smoke detection algorithm terminates at this step and outputs zero as the response.

Image Intensity Change Detection

Changes of pixel intensity values across frames indicate candidate regions containing smoke. We first enhance the contrast of image I_t, I_{t-2}, and B_t by using CLAHE (contrast-limited adaptive histogram equalization [26]) to obtain I_{heq}, I'_{heq}, and B_{heq}. CLAHE limits the contrast to avoid over-amplifying noise and operates on small local regions in the image. The desired shape of the histogram in a local region is approximately flat and follows a uniform distribution. One reason

for performing contrast enhancement is that the color and saturation of smoke may be similar to the background under some lighting conditions.

Next we perform background subtraction as shown in (3.4) on each channel of the two image pairs (I_{heq}, B_{heq}) and (I_{heq}, I'_{heq}) to obtain $S_{heq} = \text{bgSub}(I_{heq}, B_{heq})$ and $F_{heq} = \text{bgSub}(I_{heq}, I'_{heq})$, which provides information about the change of image intensity values between the current frame, background, and the previous frame. Smoothing the binary images $S_{heq} > T_3$ and $F_{heq} > T_4$ by using the process described in section 3.1.2 yields M_{heq1} and M_{heq2}.

Finally we combine M_{heq1} and M_{heq2} by using an AND operator into the resulting image M_{heq} as shown in (3.7). Figure 3.2 visualizes the steps of image intensity change detection.

$$\begin{aligned} M_{heq1} &= \text{smooth}\big(\text{bgSub}(I_{heq}, B_{heq}) > T_3\big) \\ M_{heq2} &= \text{smooth}\big(\text{bgSub}(I_{heq}, I'_{heq}) > T_4\big) \\ M_{heq} &= M_{heq1} \text{ and } M_{heq2} \end{aligned} \tag{3.7}$$

3.1.3 Texture Segmentation

Texture segmentation partitions images into regions based on their texture information. This step computes filter responses by convolving an image with a filter bank, clusters the responses into a set of textons [17], and partitions the image into separate regions by using these textons. We first combine the results of change detection algorithms by performing an AND operation on M_{dog} and M_{heq} to obtain M_{cd} as shown in Figure 3.3. If all pixel values in image M_{cd} are zero, the smoke detection algorithm stops at this step and outputs zero as the response.

Next we compute the filter bank using a variation of Laws' texture energy measures [14] as shown in (3.8).

$$\begin{aligned} L5 &= [\ 1\ \ 2\ \ 3\ \ 2\ \ 1\ ] \text{ (Level)} \\ E5 &= [\ -1\ \ -2\ \ 0\ \ 2\ \ 1\ ] \text{ (Edge)} \\ S5 &= [\ -1\ \ 0\ \ 2\ \ 0\ \ -1\ ] \text{ (Spot)} \\ W5 &= [\ -1\ \ 2\ \ 0\ \ -2\ \ 1\ ] \text{ (Wave)} \\ R5 &= [\ 1\ \ -4\ \ 6\ \ -4\ \ 1\ ] \text{ (Ripple)} \end{aligned} \tag{3.8}$$

The filter bank is a set of 5-by-5 convolution masks obtained by calculating the outer products of pairs of texture vectors in (3.8). The L5, E5, S5, W5, and R5 vectors detects gray level, edges, spots, waves, and ripples in the image respectively.

Then we take the contrast-enhanced image I_{heq}, subtract it with the mean value of I_{heq}, and convolve it with the filter bank for each RGB channel to obtain feature vectors. Each vector represents the corresponding pixel in I_{heq} in the feature space and has 125 dimensions. Then the algorithm performs Principal Component Analysis which preserves 98% of the energy (eigenvalues) on the feature vectors to reduce dimensions. Using the contrast-enhanced image alleviates the problem that some weather circumstances such as fog cause a decrease in background texture information.

Finally we perform an accelerated k-means++ algorithm [2, 7] which chooses better initialized values (seed points) to cluster the feature vectors into textons and divide the current image into various regions as shown in image R_t in Figure 3.3. Smoothing the image R_t by discarding small regions, removing noise by using a median filter, and performing morphological closing yields R_{smooth} in Figure 3.3.

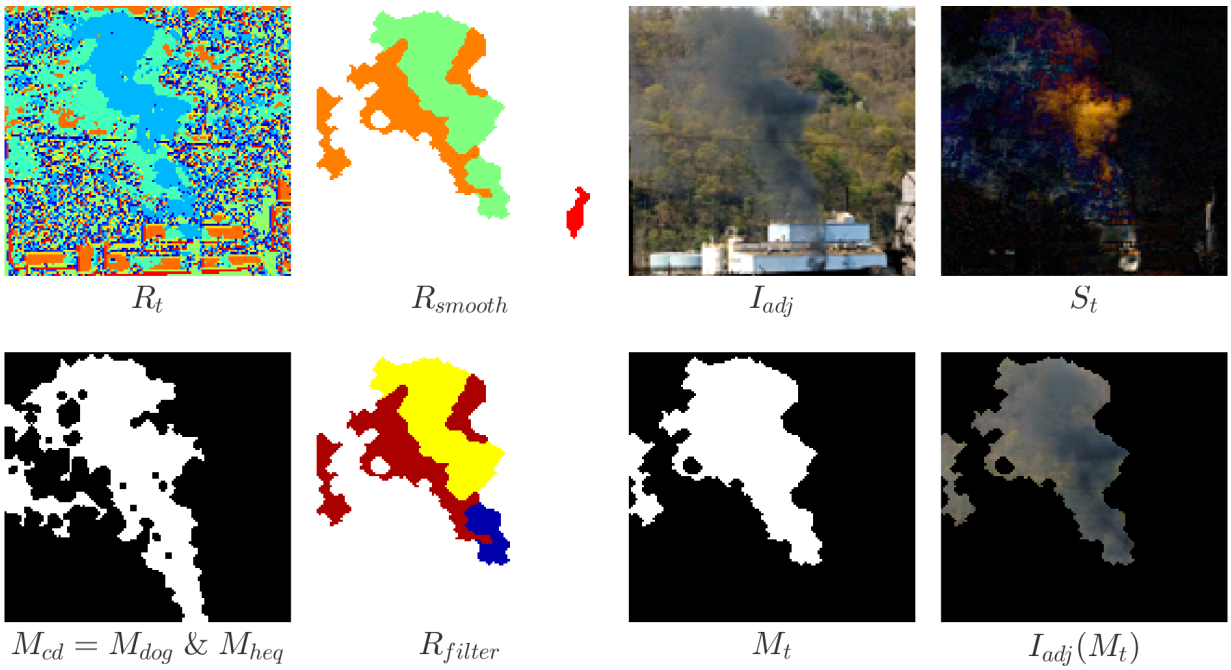

Figure 3.3: This figure demonstrates the steps of texture segmentation and region filtering. See section 3.1.3 and 3.1.4 for detailed explanation.

3.1.4 Region Filtering

Region filtering determines if a region matches the appearance and behavior of smoke by evaluating shape, color, size, and the amount of change. We first use the connected component algorithm to find all separated regions and remove the ones which are thin and narrow. Mathematically speaking, for each region, the ratio of width to height of its bounding box exceeds a certain threshold. Or the ratio of the size of the region and its bounding box is smaller than a threshold.

Next we adjust the contrast of each channel in I_t to produce I_{adj} in Figure 3.3 by stretching intensity values so that 1% of the data is saturated at low and high intensities of I_t. We group nearby white regions and black ones based on I_{adj} to reconstruct the shapes of objects. Since the color of smoke is usually grayish or bluish, we can remove regions having non-grayish and non-bluish colors described by (3.9)

$$\begin{aligned} &|c_1 - c_2| \geq t_1 \quad \text{or} \quad |c_2 - c_3| \geq t_2 \quad \text{or} \quad |c_1 - c_3| \geq t_3 \\ &c_j = \text{median}\big(I_{adj}(x, y, j)\big) \quad \forall (x, y) \in R_i \end{aligned} \tag{3.9}$$

where j indicates different channels in I_{adj}, R_i denotes the i^{th} region, (x, y) means the location of pixels, and $\{c_j : j = 1, 2, 3\}$ are the median of corresponding pixel values in R_i in the RGB channels of I_{adj}. We also remove regions having light colors described by (3.10) because steam is usually white.

$$c_1 \geq t_4 \quad \text{and} \quad c_2 \geq t_5 \quad \text{and} \quad c_3 \geq t_6 \tag{3.10}$$

Then we compute the size of each region and remove large or small ones which may be noise and shadow respectively. Furthermore, we remove the i^{th} region R_i if it does not have sufficient

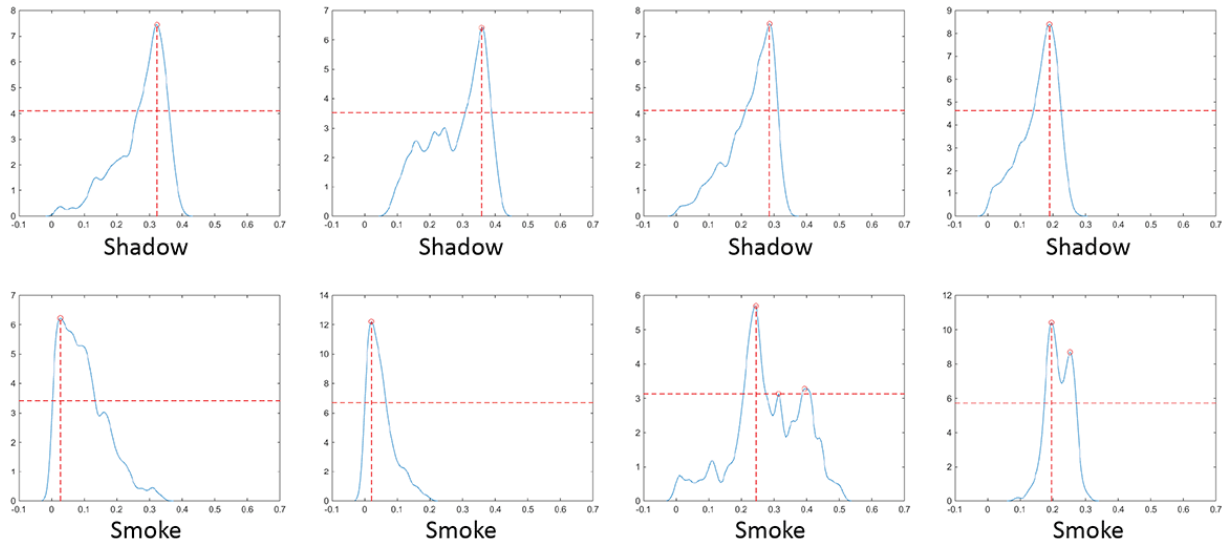

Figure 3.4: Each small graph shows the probability density function of a smoke or shadow region's corresponding pixel values in S_t (see Figure 3.3) using kernel density estimation. The x-axis represents the pixel values in S_t. The horizontal red line is the threshold for computing number of peaks. The vertical red line indicates the pixel value of the highest peak.

amount of change by summing up the corresponding pixel values in M_{cd} by using (3.11)

$$\sum_{\forall(x,y)\in R_i} M_{cd}(x,y) \leq t_7 \tag{3.11}$$

where (x, y) denotes the location of pixels in region R_i.

Finally we remove regions which may contain shadow. The algorithm performs background subtraction using (3.4) on I_t and B_t to obtain $S_t = \text{bgSub}(I_t, B_t)$ in Figure 3.3. Then we compute the probability density function (PDF) of each region's corresponding pixel values in S_t using kernel density estimation [20] with a Gaussian kernel.

$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x - X_i}{h}\right) \quad \text{where} \quad X_i \in S_t \tag{3.12}$$

Because the PDF of shadow and smoke regions have distinct characteristics (see Figure 3.4), we can describe shadow regions by utilizing (3.13)

$$\underset{x}{\text{argmax}}\ p(x) > t_8 \quad \text{and} \quad \sum_{x_i \in X} \mathbf{1}_{\{p(x_i)>t_9\}} < t_{10} \tag{3.13}$$

where x indicates pixel values, $p(x)$ is the probability density function, $\text{argmax}_x\ p(x)$ means the pixel value of the highest peak, X is a set of pixel values of the corresponding peaks, $\mathbf{1}_A$ is the indicator function of a set A, and $\sum_{x_i \in X} \mathbf{1}_{\{p(x_i)>t_9\}}$ is the number of peaks having their heights exceed a certain threshold.

Applying all the above region filtering steps on R_{smooth} yields R_{filter} (see Figure 3.3). We compute a mask M_t which is a binary image based on R_{filter} and output the response at time t as the sum of all pixel values in mask M_t.

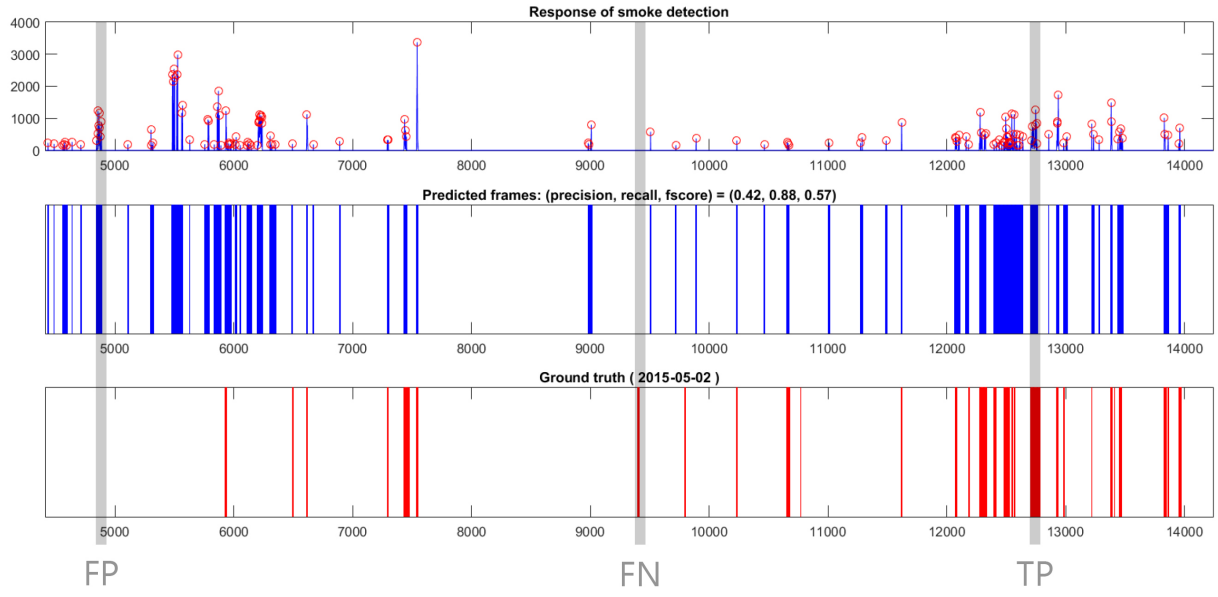

Figure 3.5: This figure shows the result of smoke detection. The x-axis and y-axis indicate the frame number and the amount of pixels identified as smoke. The bottom graph is the ground truth of May 2, 2015. The top and middle graphs show the response and the prediction of all daytime frames. The red circles in the top graph represent the local peaks. The gray bars indicate true positive (TP), false positive (FP), and false negative (FN).

3.1.5 Event Detection

Event Detection identifies the starting and ending time of fugitive emissions. We first select daytime frames for each day and ignore nighttime ones because of lighting issues. Next, we apply change detection, texture segmentation, and region filtering on these frames to obtain a time-series signal (see the top chart in Figure 3.5). Each value in the time-series signal represents the number of smoke pixels in a corresponding video frame. Then we compute segments in the time-series signal by finding peaks and corresponding peak widths. Finally we merge nearby segments into events (see the middle chart in Figure 3.5).

3.2 Experiment

We used MATLAB to develop the smoke detection algorithm and VLFeat [25] library to run the accelerated k-means++ algorithm for clustering feature vectors during the texture segmentation step. Each timelapse of a day consisted of 16838 frames. We ran the smoke detection algorithm on a window with 496-by-528 pixels in the timelapse video for 21 days in 2015 during daytime. The processing time was 30 minutes on average for a day by using all cores on a workstation with two hex-core CPU (Intel Xeon X5670).

We manually labeled these 21 days to evaluate the performance of the algorithm. The bottom graph of Figure 3.5 shows the ground truth labels on May 2. The middle and bottom graphs demonstrate the response and the prediction of smoke emissions. Table 3.1 and 3.2 show the

Table 3.1: The evaluation of all daytime frames for 9 days on May 2015.

Date	TP	FP	FN	Precision	Recall	F-score
May 1	15	36	4	0.2941	0.7895	0.4286
May 2	21	29	3	0.4200	0.8750	0.5676
May 3	24	28	8	0.4615	0.7500	0.5714
May 4	25	25	5	0.5000	0.8333	0.6250
May 5	14	19	4	0.4242	0.7778	0.5490
May 6	17	11	4	0.6071	0.8095	0.6939
May 7	26	16	3	0.6190	0.8966	0.7324
May 8	22	22	4	0.5000	0.8462	0.6286
May 9	16	23	1	0.4103	0.9412	0.5714
Avg				0.4707	0.8355	0.5964

Table 3.2: The evaluation of all daytime frames (exclude frames containing steam) for 9 days on May 2015.

Date	TP	FP	FN	Precision	Recall	F-score
May 1	13	8	4	0.6190	0.7647	0.6842
May 2	18	11	3	0.6207	0.8571	0.7200
May 3	24	19	6	0.5581	0.8000	0.6575
May 4	25	17	4	0.5952	0.8621	0.7042
May 5	13	9	3	0.5909	0.8125	0.6842
May 6	15	4	4	0.7895	0.7895	0.7895
May 7	26	6	3	0.8125	0.8966	0.8525
May 8	22	18	4	0.5500	0.8462	0.6667
May 9	14	17	1	0.4516	0.9333	0.6087
Avg				0.6209	0.8402	0.7075

Table 3.3: Evaluation of the smoke detection algorithm on 12 randomly chosen days for each month in 2015. TP, FP, and FN indicates true positive, false positive, and false negative respectively.

Date	TP	FP	FN	Precision	Recall	F-score
Dec 22	18	21	7	0.4615	0.7200	0.5625
Nov 15	18	6	1	0.7500	0.9474	0.8372
Oct 05	27	23	0	0.5400	0.9643	0.6923
Sep 09	10	35	8	0.2222	0.5556	0.3175
Aug 13	28	35	2	0.4444	0.9333	0.6022
Jul 08	15	35	9	0.3000	0.6250	0.4054
Jun 11	22	14	4	0.6111	0.8462	0.7097
May 28	24	17	3	0.5854	0.8889	0.7059
Apr 02	15	28	10	0.3488	0.6000	0.4412
Mar 06	1	8	15	0.1111	0.0625	0.0800
Feb 10	3	32	10	0.0857	0.2308	0.1250
Jan 26	1	5	2	0.1667	0.3333	0.2222

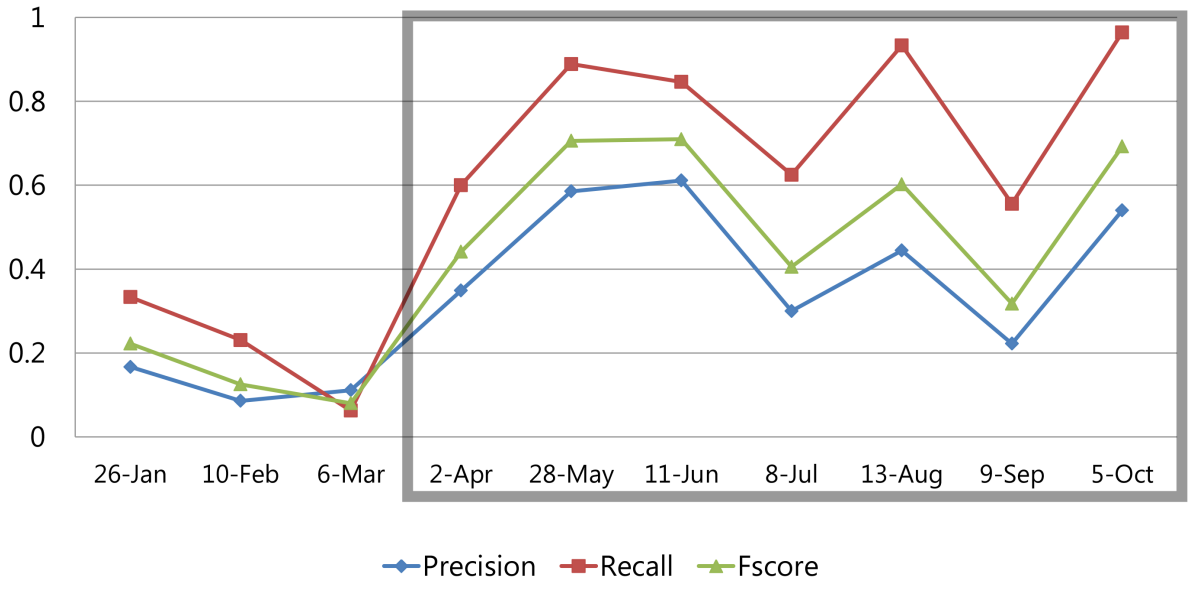

Figure 3.6: Evaluation of the smoke detection algorithm on 12 randomly chosen days for each month in 2015.

results of evaluation from May 1st to 9th with and without the frames having steam. Table 3.3 shows the accuracy of twelve randomly picked days for each month in 2015.

We calculate true positives (TP), false positives (FP), false negatives (FN). Denote the boolean array of ground truth labels G and predictions P which contains only true and false entries. We first group the continuous true entries in G into a series of segments and apply the same process on prediction P. Next, for each segment in P, denote the starting and ending frame indices m_p and n_p. We mark a segment as a true positive if 30% of the entries in the segment contains true ground truth labels, which is described in (3.14). Otherwise, we mark the segment as a false positive.

$$\frac{\sum_{i=m_p}^{n_p} G(i)}{n_p - m_p + 1} > 0.3 \tag{3.14}$$

For each segment in G, denote the starting and ending frame indices m_g and n_g. We mark a segment as a false negative if $\sum_{i=m_g}^{n_g} P(i) = 0$, which means no entries in the segment contains true predictions. Finally, we compute precision (PR), recall (RE), and F-score by using (3.15).

$$\begin{aligned} \text{PR} &= \text{TP}/(\text{TP} + \text{FP}) \\ \text{RE} &= \text{TP}/(\text{TP} + \text{FN}) \\ \text{F-score} &= 2 * \text{PR} * \text{RE}/(\text{PR} + \text{RE}) \end{aligned} \tag{3.15}$$

3.3 Visualization

We apply the smoke detection algorithm on the nine months in 2015 with better accuracies (see the highlighted rows in Table 3.3 and Figure 3.6). We visualize the results by using an interactive timeline, an autonomous fast-forwarding function, and a visual summary (see Figure 3.7 and Figure 3.8). The programing languages of the visualization are JavaScript and HTML. The interactive timeline serves as an indicator; community members can click on a spike on the timeline to seek to a frame with fugitive emissions. The fast-forwarding function plays only the smoke-sensed time segments in the timelapse video and skips the rest. The visual summary provides a collection of animated images generated from the time segments. Each image has a hyperlink to the timelapse viewer at a particular time and view. Community members can share or collect desired smoke images in online documents as visual evidence.

Figure 3.7: This figure shows the timelapse viewer with an interactive timeline which visualizes the results of smoke detection and provides indicators for seeking to interesting events. The fast-forwarding button only plays interesting segments and skips the rest. The image button gives a collection of animated images (see Figure 3.8) which are likely to contain smoke.

Figure 3.8: This figure shows a part of the collection of animated images generated by the timelapse viewer according to the results of smoke detection on May 2, 2015. The local community can select desired images and drag them into a Google Doc for documentation, presentation, and storytelling. The hyperlinks under the animated images redirect users back to the timelapse viewer upon mouse clicking.

Chapter 4

Discussion and Future Work

The system significantly reduces the workload and the time of searching and documenting fugitive emissions so that local community members can focus on the content of documentation instead of laborious and time-consuming works. It takes approximately 840 seconds to play the timelapse video at 12 frames per second for a day. With the fast-forwarding feature, the local community members can now browse candidate frames containing smoke for a whole day in a shorter time. Moreover, manually searching and generating animated images of smoke emissions through all daytime frames takes about 1.5 hours. The viewer can now provide a collection of images as shown in Figure 3.8 almost instantly when citizen scientists click the image button on the user interface.

The system has several limitations. First, the algorithm uses a heuristic approach. There are numerous thresholds that one needs to adjust for the algorithm. We use only one set of tuning parameters. It is an open question as to whether the same parameter set works for daytime frames in other days. In addition, since the lighting conditions of nighttime and daytime are different, detecting smoke emissions at nighttime may require another algorithm.

Second, the system produces a large portion of false positives caused by steam or fast-moving shadows and a small portion of false negatives caused by low opacity or small-sized smoke. The algorithm has difficulties in completely excluding fast-moving shadows due to the combination of wind and partly cloudy conditions. It also cannot distinguish between white smoke and steam clearly. These are the reasons why the precisions in table 3.1 are low. The overall accuracy increases if we remove the frames having steam (see Table 3.2). We use frame differencing (see section 3.1.2) to remove steam regions, which assumes three conditions: (1) the size of steam is large; (2) wind blows at a constant direction slowly; and (3) steam emits from places that are inside the detection window. Nevertheless, steam may have low opacity or change a lot across multiple frames due to high wind speed, which makes it numerically similar to white smoke (see Figure 1.2). Currently we rely on citizen scientists to exclude steam using human vision from the presented collection of animated images.

In the future, we plan to generalize this work by obtaining more labels using crowdsourcing, turning smoke images and these labels into a reusable dataset, and training a classifier using these labels.

Bibliography

[1] Timelapse Viewer. http://timemachine.cmucreatelab.org/. 1

[2] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. 3.1.3

[3] Simone Calderara, Paolo Piccinini, and Rita Cucchiara. Smoke detection in video surveillance: A mog model in the wavelet domain. In *Computer Vision Systems*, volume 5008 of *Lecture Notes in Computer Science*, pages 119–128. Springer Berlin Heidelberg, 2008. 2

[4] Turgay Çelik, Hüseyin Özkaramanli, and Hasan Demirel. Fire and smoke detection without sensors: Image processing based approach. In *European Signal Processing Conference*, pages 1794–1798, 2007. 2

[5] Sen-Ching S. Cheung and Chandrika Kamath. Robust background subtraction with foreground validation for urban traffic video. *EURASIP J. Appl. Signal Process.*, 2005:2330–2340, January 2005. 2

[6] Robert Collins, Alan Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, and Osamu Hasegawa. A system for video surveillance and monitoring. Technical report, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2000. 2

[7] Charles Elkan. Using the Triangle Inequality to Accelerate K-Means. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 2003. 3.1.3

[8] Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, UAI'97, pages 175–181, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. 2

[9] Jayavardhana Gubbi, Slaven Marusic, and Marimuthu Palaniswami. Smoke detection in video using wavelets and support vector machines. *Fire Safety Journal*, 44:1110 – 1115, 2009. 2

[10] Muki Haklay. Citizen science and volunteered geographic information: Overview and typology of participation. In *Crowdsourcing Geographic Knowledge*, pages 105–122. Springer Netherlands, 2013. 1

[11] Simon Philipp Hohberg. Wildfire smoke detection using convolutional neural networks.

Technical report, Freie Universitt Berlin, Berlin, Germany, September 2015. 2

[12] I. Kopilovic, B. Vagvolgyi, and T. Sziranyi. Application of panoramic annular lens for motion analysis tasks: surveillance and smoke detection. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 714–717 vol.4, 2000. 2

[13] Robert Kosara and Jock MacKinlay. Storytelling: The next step for visualization. *Computer*, 46(5):44–50, 2013. 1

[14] Kenneth I. Laws. *Textured Image Segmentation*. PhD thesis, University of Southern California, Los Angeles., Jan 1980. 3.1.3

[15] Chen-Yu Lee, Chin-Teng Lin, Chao-Ting Hong, and Miin-Tsair Su. Smoke Detection Using Spatial and Temporal Analysis. *International Journal of Innovative Computing, Information and Control*, 8:4749–4770, 2012. 2

[16] Kwan Liu Ma, Isaac Liao, Jennifer Frazier, Helwig Hauser, and Helen-Nicole Kostis. Scientific storytelling using visualization. *Computer Graphics and Applications, IEEE*, 32(4): 12–19, 2012. 1

[17] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43:7–27, 2001. 3.1.3

[18] R.J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. *Image Processing, IEEE Transactions on*, 14(3):294–307, March 2005. 2

[19] Randy Sargent, Chris Bartley, Paul Dille, Jeff Keller, and Illah Nourbakhsh. Timelapse GigaPan: Capturing, Sharing, and Exploring Timelapse Gigapixel Imagery. In *Fine International Conference on Gigapixel Imaging for Science*, 2010. 1

[20] Bernard Walter Silverman. Density estimation for statistics and data analysis. In *Computer Vision Systems*, Number 26 in Monographs on Statistics and Applied Probability. Chapman & Hall, 1986. 3.1.4

[21] Jonathan Silvertown. A new dawn for citizen science. *Trends in Ecology & Evolution*, 24 (9):467–471, 2009. 1

[22] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 1999. 2

[23] Hongda Tian, Wanqing Li, Philip Ogunbona, editor="Cremers Daniel Wang, Lei", Ian Reid, Hideo Saito, and Ming-Hsuan Yang. *Single Image Smoke Detection*, chapter Computer Vision – ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part II, pages 87–101. Springer International Publishing, 2015. 2

[24] B.U. Toreyin, Y. Dedeoglu, and A.E. Cetin. Wavelet based real-time smoke detection in video. In *Signal Processing Conference, 2005 13th European*, pages 1–4, Sept 2005. 2

[25] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008. http://www.vlfeat.org/. 3.2

[26] Karel Zuiderveld. Contrast Limited Adaptive Histograph Equalization. pages 474–485. Academic Press Professional, Inc., 1994. 3.1.2