A System for Determining Indoor Air Quality from Images of an Air Sensor Captured on Cell Phones

Kelsey Whitesell
Center for Embedded Networked Sensing
UCLA 3563 Boelter Hall
Los Angeles, CA 90095-1596
(310) 206-2476
Kelsey.whitesell@colorado.edu

Brendan Kutler
Center for Embedded Networked Sensing
UCLA 3563 Boelter Hall
Los Angeles, CA 90095-1596
(310) 206-2476
newruralsociety@gmail.com

Nithya Ramanathan, Deborah Estrin
Center for Embedded Networked Sensing
UCLA 3563 Boelter Hall
Los Angeles, CA 90095-1596
(310) 206-2476
nithya@cs.ucla.edu, destrin@cs.ucla.edu

ABSTRACT

We have designed and implemented a system that analyzes images of air sensors taken on cell phones and extracts indoor air pollution information by comparing the sensor to a calibrated color chart. This system will be deployed with Project Surya, a pilot project of the UN Environmental Program that aims to replace traditional cooking methods with clean cooking technology in several rural Indian villages. In order to evaluate the effectiveness of the cooking technology at reducing indoor pollution, several hundred homes will receive a special air sensor that changes brightness based on the concentration of air pollution. Our system is capable of automatically getting data from multiple sensors each day by analyzing photographs of them. In order to operate in the presence of noise and inconsistent lighting conditions, our system employs a median sensor, finds the light intensity gradient across images, and calibrates the images if possible. We used a GVF active contour segmentation algorithm that can segment the two objects in the images (the sensor and the color chart) followed by a mode brightness matching algorithm to match the sensor and color chart. We tested our system by comparing results of manual analysis of seven different sensors to results obtained using our system. In these tests, our system was shown to match the sensor to a color on the chart with a maximum discrepancy of 1 bar on the color chart from the average result of manual analysis, with an average deviance of .406 of a bar.

Categories and Subject Descriptors

D.2.10[**Software Engineering**]: Design – *Methodologies;* I.4.6**[Image Processing and Computer Vision]**: Segmentation – *Region growing, partitioning*; I.4.8J.m**[Computer Applications]**: Miscellaneous

General Terms

Algorithms, Measurement, Performance, Design, Reliability,

Keywords: cell phones, image analysis, calibration, active contour segmentation, automation, air sensors

1.INTRODUCTION

Project Surya plans to deploy clean-cooking technology in an attempt to reduce air pollution in rural Indian homes. Most stoves currently used in rural India burn biomass, like wood, which release harmful pollutants. Surya focuses on black carbon, commonly known as soot, because of its immediate, negative impact on health and climate [4].

In order to gage the effectiveness of their cookers, climate scientists at Project Surya have designed a gray, chemical sensor that changes brightness depending on the concentration of black carbons (BCs) in the air. When implemented, the sensor will be attached to a battery-powered pump that draws air through the sensor [4]. In this paper, this sensor will be referred to as the sensor to avoid confusing it with image sensors. Project Surya has also designed a calibrated grayscale color chart with of nine bars. Each bar is a different brightness and corresponds to a different BC concentration. Thus, BC concentration can be determined by matching one of the bars on the chart to the brightness of the sensor. The color chart and several sample sensors that have been removed from the pump are shown in *Figure 1.* Matching the brightness of a sensor to a color chart would usually be done manually by a person. However, Project Surya's operations will be large, deploying cookers in 6,500 homes in the pilot phase alone [1]. Furthermore, every day

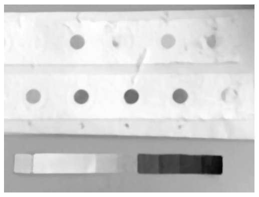

Figure1. A set of sample sensors and the color chart shown after the median filter has been applied

data must be extracted from the sensors and the sensors replaced. Relying on people to read the sensors is fraught with problems: a high probability that someone would forget to record data or replace the sensor, results biased by local lighting conditions, or failure to account for certain conditions, such as light variation and browning (see *Section 2.2*).

We propose a robust image analysis system to collect accurate data from the sensors. Automated computer systems are amenable to matching since this task is tedious and objective. Our system automatically compares the brightness of the sensor to the co-located color chart in an image and outputs the BC concentration. In our system, brightness refers to pixel brightness, or the average of the RGB pixel values. Images will be uploaded from N80 or N95 cell phones to a central server, where the analysis will be performed. All raw data and the results of analysis will be stored in a central database.

Our system has to deal with a number of faults introduced by the uncertain deployment environment. Our primary goal was to design the system to either fix or register all faults and delivering straightforward, accurate output. Steps getting accurate output from this system include assessing image quality, sensoring out noise, and calibrating images to account for lighting conditions. Noise is further discussed in *Section 2.2*. Inconsistent lighting conditions within images can easily affect perceived brightness, the only quality available to use when matching the sensor to a grayscale color chart. Variation in lighting conditions *between* images is not a problem because the sensor and the color chart are always to be photographed in the same frame. Within individual images, the system must determine how light varies across the image. Variations can include light fading across the image, concentrated sunlight, or the camera's flash, which can discolor images if the camera is held too close to the sensor. Our system determines inconsistent lighting across images and adjusts the images so that the ambient light across the sensor is the same as that across the color chart.

In situations where the image quality is too compromised, the image is discarded and the participant is requested to take another image. Our system is also robust to intermittent Internet connections.

2.METHODS AND DESIGN

Remote analysis of the air sensor begins with a resident of a home containing a sensor taking a picture of the sensor and color chart in the same frame. If network connectivity is available, the phone uploads the image to the database. We used SensorBase, an online database for the Center of Embedded Networked Sensing (CENS) at UCLA. A daemon queries SensorBase every three hours and analyzes up to 1221 images. Query period and restraints on the number of images accepted during each period are discussed in *Section 3.2*.

For each image uploaded to the database, image quality is assessed, and the image is calibrated and segmented. Finally, the brightness of the sensor is matched to the color chart. The original image, any modifications or errors, and the results of the analysis are uploaded to SensorBase. An SMS message is sent back to the participant with information on pollution in their home [5]. The methods used to accomplish each of these steps are detailed below**.**

2.1Segmentation

Our system uses a GVF active contour, or snake, to identify the sensor and color chart. We considered several different algorithms. Normalized cuts depend too heavily on choice of parameters to be totally effective in automated systems. GVF active contours, which are based on models of physical forces called Gradient Vector Fields (GVF) have been adapted to successfully segment multiple objects without requiring manual input [8],[9]. Our system uses an automated version of the deformable, GVF snake model developed by C. Xu and J.L. Prince [9]. This method segments the color chart into separate bars, returning mathematical definitions of the nine bars on the scale. This mathematical definition is a list of (x,y) points in the image that bound each object.

We are currently looking to reduce the time to run of our segmentation algorithm and obtain a more easily parse able output by our system's segmentation by coupling the GVF active contour with algorithms in python from Eric Debreuve's Matlab's Active Contour Toolbox. The most important change would be using a blind "uniform" initialization that involves initializing multiple contours and deforming them into separate objects [10].

2.2 Image Quality Analysis and Handling of Low-Quality Images

When an image is uploaded to the server, the system processes it to check for problems and fixes any faults possible. It begins by correcting discolored images. We found that discoloration occurs when the cell phone camera flash floods the image. This phenomenon was observed when the cell phone camera was held too close to the sensor and chart. We found that this discoloration was fairly uniform across each image, so that the brightness of the chart and the senor were affected to the same degree, negating the need for calibration. Instead, our system simply converts discolored images to grayscale and continues analysis.

Noise was found to be a problem in several images. Noise included random specs on the image, caused by dust, dirt, or small bugs settling on the sensor. In order to remove noise from an image, the system applies a median filter from the Python Imaging Library[6]. The filter size is nine. This means the sensor sensors using the median of the pixels in a 9X9 neighborhood around each pixel. This size was chosen because we found it filtered the same amount of noise as larger sizes while requiring less time to run. This filter effectively eliminated noise except in cases where the noise occupied more than 15 percent of the image. After receiving data from Surya's pilot program, we plan to develop a common noise model and examine how well our system eliminates noise following that model.

The system also checks the image resolution, discarding images with a resolution of less than .25 megapixels. This number was chosen based on the minimum number of pixels needed to accurately calculate mode brightness, 2500 square pixels(see *Section 3.2*), and the minimum percentage of the image that the sensor is likely to occupy. We predicted minimum percent occupancy by having six people photograph a sample sensor and color chart as if they were participating in Surya's pilot study. In these photographs, the sensor never occupied less than ten percent of the entire image. This implies that the image must have a resolution of at least .25 Mpixels in order to obtain an accurate result even when the sensor occupies only ten percent of the image.

To account for variable lighting conditions, our system finds the light intensity gradient function, an expression of how pixel brightness is changing across the image. This gradient reflects how the ambient light changes across the image. We calculate light intensity gradient in the horizontal and vertical directions by taking the partial derivatives of an interpolated function, *L(x,y),* of pixel brightness across the image. The brightness function is interpolated using the SciPy library for python. There are three cases: 1) the gradient is always zero, 2) the gradient is never zero, and 3) the gradient is zero for a specified region surrounding the sensor and the color chart. Only in the third case are the images able to be calibrated.

If $\frac{\partial L}{\partial x}$ and $\frac{\partial L}{\partial y}$ equal to zero over the entire image, the image is entirely one brightness. Such images cannot be analyzed because it is impossible to differentiate between objects in the image.

If $\frac{\partial L}{\partial x}$ and $\frac{\partial L}{\partial y}$ are not zero over the entire image, our system might be able to calibrate the image. Calibration involves tweaking the brightness of the sensor until it is as if the sensor was experiencing the same ambient lighting conditions as the color chart when the image was captured. This is achieved using the brightness of a set region R surrounding the color chart as a "base brightness," B_b. Let d be the difference between the B_b and the brightness of the sensor in the image. It then follows that if the brightness surrounding the color chart in region R was the same as B_b. as would the case if the ambient lighting conditions were the same across the sensor and the color chart, the brightness of each bar on the chart would be its brightness in the image plus d. This calibration algorithm was used because of its straightforwardness and low computational demands. A more complicated algorithm is not needed for images containing such simple shapes.

If $\frac{\partial L}{\partial x}$ and $\frac{\partial L}{\partial y}$ are never zero across the image, ambient lighting conditions are variable across the entirely image. This renders the above calibration impossible, because there is no region R around the sensor for which brightness is constant. However, if $\frac{\partial L}{\partial x}$ and $\frac{\partial L}{\partial y}$ are zero in region R, then lightning conditions are constant in R, B_r can be calculated, and the image can be calibrated as described above. In our system, R has a width of 50 pixels. The reasons for choosing this value are given in *Section* 3.2.

Using the region around the chart as the calibration base standard eliminates allows the system to calibrate only the single sensor object rather than every bar on the color chart.

Our system does not address several faults we are aware of. As can be seen in *Figure 1*, some wrinkles in the material the sensor is made are still visually present after the median sensor is applied. Wrinkles in the paper cast shadows and cause the system to register a light intensity gradient, rendering it unable to calibrate the image if the wrinkles exist in R. Currently, our system has no computational way to account for these wrinkles.

Another problem not addressed is browning. Browning of the sensor has been observed when there is an excessive amount of dirt or dust in the air. We do not address this problem because Project Surya is currently looking to develop a physical mechanism to remove brown particles [4].

Lastly, our system assumes that the image contains both the sensor and the color chart. If it does not, the system will try to match the sensor to a random spot in the image. In most cases, this could be fixed by adding a program that registers an error if two objects are not returned from segmentation.

2.3 Brightness Matching

Once the quality of the image has been verified and the any necessary calibration performed, the sensor brightness can be matched to the color chart. We chose to match the mode brightness of a 50x50 pixel region (see *Section 3.2*) within the sensor to the mode brightness of each bar on chart. The mode was chosen over the average or median brightness in case random noise is not completely eliminated by the median filter, as was found to occur for images where the sensor occupies a high percent of the image.

In order to match the brightness of the sensor to the chart, the system first needs to identify the separate bars in the color chart and the corresponding BC concentration. The system obtains only coordinates defining the boundaries of each bar during the segmentation phase. Because of this, we chose a boundary-based shape orientation algorithm developed by Jovisa Zunic and Milos Stojmenovic to orient each bar on the color chart. This method has been shown to

Table 1. System Parameters

Parameter	Description	Sub-system	Current Value
Δ	# of images processed in each query period	Image Slogging	1221
ITER1	iterations computed for the snake deformation	Segmentation	8
ITER2	Iterations computed for the GVF	Segmentation	100
R	Radius of region used for calibration	calibration	50 pixels
r	Dimension of the region used to calculate sensor brightness	brightness matching	50 pixels

be effective at orienting polygonal shape when only the boundary conditions are known.

After the individual bars oriented, the system finds the bar with the smallest minimum x-coordinate. If all the bars have the same minimum x-coordinate, then the chart is positioned vertically and the mode brightness of the bar with the smallest minimum y-coordinate is calculated. Because the orientation of the chart is straight and so limited to the arrangements where inner bars can never be further to the left than the end bars, that bar should be either the black bar or the white bar. This assumes that the sensor is left of the sensor in the image. The brightness is calculated for the pixels within the bounds defining that initial bar. If the bar is black (brightness ≈ 0), that initial bar is bar 9 (*Figure 2*). If the bar is white (brightness ≈ 255), the bar is bar 0. The numbers the other bars are found using the location and number of the initial bar. The brightness within the bounds of each bar is then measured and associated with the correct number.

Once the chart is oriented, the mode brightness of sensor can be matched to a bar on the color chart using a simple comparison loop. When a match is found, information about what that bar indicates about the BC concentration is recorded in SensorBase and sent to the cell phone.

2.4 Implementation

Our system was written entirely in Python because Nokia N80 and N95 cell phones can be programmed using Pys60; thus, we were able to run code that uploads images to the SensorBase on the phone. The code currently on the server could also easily be translated onto the phone if such a transfer were found to be desirable.

3.EVALUATION

3.1 Resource Utilization

Our system normally takes 8.842 seconds to analyze one image, but can take up 2.75 minutes to run if analyzing high resolution images with a slow server response. Run time for each individual process is detailed in *Table 2*.

Table 2. Time to Run

Process	Time to Run (seconds)
Sensorbase Image Slogging	Normal Response: 0.135 with slow server response: 24.118
Segmentation (Low Res)	4.142
Snake Analysis (High Res)	136.762
Other Image Processing	4.565

3.2 Sensitivity Analysis

Table 1 lists the adjustable parameters in our system.

Δ is bounded at 1221because best run time is 8.842 seconds, so server analyzing images after three hours when new images are uploaded if Δ is greater than 1221 and become backlogged. This means that our current system can analyze up to 9,768 images per day. This meets Surya's need to analyze 6,500 images per day. However, a slowest run time it cannot analyze more than 480 images per day. The three hour query period can be adjusted as desired as long as time to run is taken into account to avoid backlog.

Our ITER1 and ITER2 values are higher than those chosen by Xu and Prince in their implementation of the GVF snake because pictures taken on N80 and N95 cell phones have a higher resolution than those tested by Xu and Prince and so require more iterations to complete the segmentation process. The current ITER1 and ITER2 properly segment a .25Mpixel image. ITER1 and ITER2 must be increased proportionally to the resolution of the image. R and r are set at 50 pixels because our tests showed that r can be between 50 and 500 pixels for images with a maximum deviation in the mode brightness of four units. Such a small deviation would not affect which bar our system matched to the sensor because the difference in brightness between bars on the color chart is always greater than 4. R may be increased so long as the light intensity gradient is still zero everywhere within R. r can be as large as the sensor radius and still return accurate results. We also found that the shape of the sensor region analyzed does not affect results.

3.3 Accuracy

Accuracy was determined by comparing results of our system to those obtained when six people visually matched the chart to a sensor. It is difficult to say if human vision is truly the most accurate ground truth for matching images based on brightness, since every person's vision is slightly different.

To test the legitimacy of this ground truth, we gave seven sensors and a copy of the color chart to six people and instructed them to match each sensor to the color chart. *Figure 3* shows the results of this study. This data supports the choice of human vision as a ground truth, but also suggests that our system would be beneficial in reducing subjectivity while sill returning results accurate with respect to those obtained using human vision. We also compared the results of manual analysis with the results from our system. This system matched the sensor to the same bar as at least one person 99 percent of the time, and was found to have a maximum deviance of one bar from the average result of our chosen ground truth.

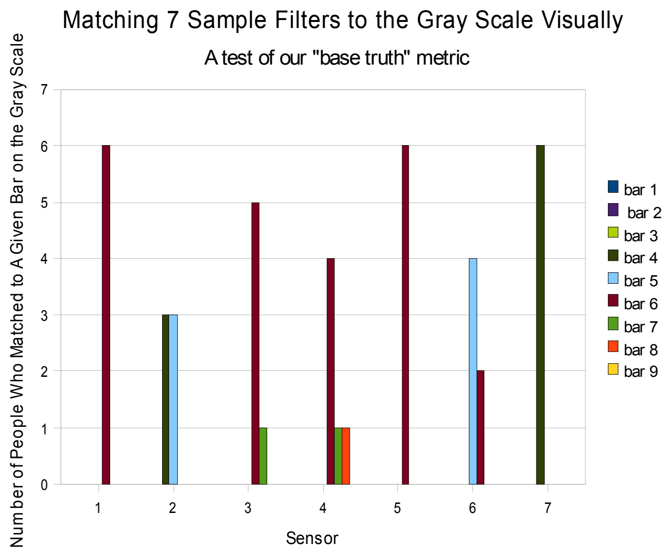

Figure 3. Test of our ground truth

4. FUTURE WORK

Further work is needed in the testing of this system. Because lack of contrast makes it difficult for the snake to construct an accurate edge map, more tests should be run segmenting images where the sensor brightness is very close to the brightness of the paper the sensor is mounted on. Again, we are looking into improving segmentation by applying concepts of blind, uniform initialization. Decreasing maximum run time would also guarantee that data from all of Surya's sensors could be analyzed daily. Parallel computing is a possible answer to this problem.

5. ACKNOLEDGEMENTS

Thanks to Summer@CENS for providing us with the resources to undertake this research.

6. REFERENCES

[1] Ramanathan, V. and K. Balakrishnan. "Project Surya: A White Paper." San Diego, Scripps Institute of Oceanography, U. of California; Chennai, India, Sri Ramachandra Medical College and Research Institute, 2007.

[2] Smeulders et. al. "Content-Based Image Retrieval at the End of the Early Years." IEEE Transactions on Pattern Analysis and Machine Intelligence. 22.12(Dec. 2000): p1349-1380.

[3] Flickner et. al. "Query by Image and Video Content: The QBIC System." Computer: the Flagship Publication of the IEEE Computer Society. 28.9(Sept. 1995): p23-32.

[4] Meeker, Karen. Untitled Thesis. UCSD, 2008.

[5] Meguerdichian et. al. "Project BudBurst: An Application of SMS and MMS to Citizen Science and Participatory Sensing." UCLA, 2008.

[6] Python Imaging Library Handbook.2005. Diango. 5 July 2008 <http://www.pythonware.com/library/pil/handbook/imagesensor.htm>

[7] Sezgin, Mehmet and Bulent Sankur. "Survey over image thresholding techniques and quantitative performance evaluation." J. Electronic Imaging.13.1(1 March 2004):

[8] Chuang, Cheng-Hung and Wen-Nung Lie. "Automatic snake contours for the segmentation of

multiple objects." Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium. 2(6-9 May 2001):p389-392.

[9] Xu, Chenyang and J.L. Prince. "Gradient vector flow: a new external force for snakes." ComputerVision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on. (17-19 Jun 1997): p66-67.

[10] Active Contour Toolbox. 27, June 2008. The MathWorks, Inc. 22 August 2008. <http://www.i3s.unice.fr/~debreuve/acontour.htm>

[11] Jovisa Zunic and Milos Stojmenovic. "Boundary based shape orientation." Pattern Recognition. 41.5(May 2008): p1768-1781