PhoneGuard: A smartphone in the coalmine

Nicolas Mayor
Mobile Systems
University of Applied Sciences of Western Switzerland
Fribourg, Switzerland
Jean-Frederic.Wagen@hefr.ch

Jean-Frederic Wagen
Mobile Systems
University of Applied Sciences of Western Switzerland
Fribourg, Switzerland
Jean-Frederic.Wagen@hefr.ch

Abhishek Samanta
College of Computer and Information Science
Northeastern University
Boston, USA
samanta@ccs.neu.edu

Ravi Sundaram
College of Computer and Information Science
Northeastern University
Boston, USA
koods@ccs.neu.edu

***Abstract*— In the short span of less than a decade the mobile phone has become a ubiquitous feature of life in India. Everyone from the chai-wallah to the CEO has a cell-phone and in recent years many of these are smart-phones capable of running smart-apps. Our focus is on conceiving a practical and useful app that can aid in disaster prevention and actually implementing and testing it. Towards this end we created PhoneGuard – a smart-app that converts the phone into a remote monitoring device – you leave the phone in a sensitive location like on the banks of a river or inside a coal-mine and it periodically takes pictures, does simple image analysis and checks for coherence over time, if warranted (e.g. in case of flooding of the river or buckling of beams in a coal-mine) it raises an alarm and follows an escalation procedure to push live images to a webpage. A complete working system was developed using standardized software and tested using realistic conditions. We believe that the image analysis and its integration into an escalation procedure is a novel aspect of our system.**

Wireless networks, sensor networks, mobile networks, Wi-Fi, cellular networks, smartphones, pattern-recognition, image-recognition,

I. Introduction

A. Motivation

It was tragic to hear that as recently as last July 29, 2011 a family of five in Central India went on a picnic to Patalpani near Indore city in central India when flash floods swept them in a trice. News of death of humans and livestock in flash floods are so routine, particularly in Bihar, that even media, both print and TV, pays no more than nominal attention. However, even flash floods have to have a beginning and had information on changes to the topography of upper reaches of water flow been available the family would have been saved from going to a watery grave.

Similarly floods and wall and roof collapse in underground mines are frequent. Here too, early processes leading to collapse, if captured in time, could save lives. The mining community in India can never forget the Chasnala Mine disaster in Bihar where 375 lives were lost.

Technology advances in communication can today improve our ability and skills to improve the disaster preparedness of the community, provide early warning to all, track hazards and timely alert to vulnerable sections to move to safer places with essentials. Any technology used for such purposes should be, as simple as possible, easily comprehensible, open and locally responsive. Such technologies as National Disaster Warning System quite often uses advanced and sophisticated technologies leading to delivery of alerts to specialists and designated officials first before dissemination to those who are the nearest to the loci of the disaster. Effectiveness of disaster preparedness systems should have in built capacity to receive monitoring information, process changes as they occur on a real time basis and convert results of such processing to credible and timely alert to simple folks in their own language.

The one obvious choice of technology is the ubiquitous mobile phone which, nowadays in India, even village folks are getting used to, staying in touch with near and dear ones across the country. In fact, the motivation for this project came from an observation made by one of the authors when recently visiting Bhedaghat [12], a beautiful location on the banks of the Narmada River near Jabalpore, Madhya Pradesh. Bhedaghat is famous for the colorful marble cliffs that flank the river banks and tourists take boat cruises to admire them. The author was chagrined to discover that boats were not plying the river till October because the boatmen were afraid of the sudden surges in the level of the river caused by the release of water from the dam up-stream.

The boatmen were confronted with the Hobbesian choice of endangering their own and their customers' lives or giving up their livelihood and facing poverty. At the same time the author observed that almost all the idle boatmen had cell phones, some of them even smart phones. Thus was born the idea of creating an app whereby a boatman could leave a smart phone upstream and automatically get notified, in advance, of a water level increase so that he could get back to shore (or prepare otherwise) for the impending surge. Such an app could also be utilized for other disaster notifications such as fires, floods, building/bridge collapses etc. The app would be of particular value where the monitoring has to be done on a one-off basis, is localized and small-scale (requiring inexpensive solutions) and underground or indoors (hence inaccessible to satellites) .

B. *Related Work*

There has been much work on the use of wireless sensor networks for disaster management. A wireless sensor network consists of autonomous sensors that are spatially distributed. The sensors can be used to *monitor* physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to a main location. Wireless sensor networks are an intensive area of research with numerous workshops and conferences arranged each year. Entire books have been written on the subject [1, 2]. [3] focuses on network protocols for the collection of information (regarding earthquakes) from a network of sensors. [6] designs a sensor network for the detection of gas leaks from pipelines. [4] studies the use of sensor network for ongoing monitoring and assessment of underwater pollution in the oceans. A common property of these various studies is the use of specialized sensor nodes. However there are genuine drawbacks with using networks of specialized sensor nodes. They require extensive maintenance. When batteries run out then the nodes need to be accessed and the batteries replaced. But most importantly they require a considerable amount of initial capital investment to get the basic infrastructure off the ground. For these (and other) reasons wireless sensor networks have not really gotten off the ground. The fact that even today there is no large scale commercial deployment is testimony to the drawbacks of this style of infrastructure development. By contrast the uptake and growth of cellular infrastructure (in a country like India) has been remarkable [11]. Our focus is therefore on using off-the-shelf infrastructure such as the cellular network and the use of mobile devices such as smart phones. In the context of smart phones and personal devices such as tablets there has been tremendous development in the commercial world in the areas of entertainment (e.g. games such as "Angry Birds") and personal productivity (e.g. social networking apps). However, due primarily to a lack of natural business models there has been less effort devoted to the development of apps in the area of disaster management. There are a number of apps for remote monitoring such as baby monitors and pet monitors [5]. However these apps are little more than webcams that remote the audio and video signal captured from a remote location. Our app is different in that it does image processing using the (limited) on-board computational capabilities of the device and then generates alarms based on an escalation procedure and the device's assessment of the situation.

C. *Our Contributions*

Our main contributions are as follows:

- We conceived of an essentially standalone app that can transform a smart phone into a tool for disaster prevention. Unlike complex solutions involving wireless sensor networks and satellites, ours is a simple cheap off-the-shelf solution that can be easily utilized by an individual for preventing even small-scale localized crises and disasters.
- The smart phone has many sensing capabilities – audio, video, motion-detection etc. We focused on the video aspect since image analysis is very compute-intensive and typically considered to be excessive for smart phones. We identified a key attribute of many disaster scenarios where the base image is relatively static and the advent of the disaster is characterized by a sudden and radical transformation in the base image. Even though actual image analysis (e.g. identifying individuals or features) is difficult we show that radical transformations in the base image can easily be detected even with the limited on-board compute-capabilities of the smart phone – we term this Coherence Disruption Detection (CDD). We believe that the realization that in the common case it is adequate to track the coherence, or lack thereof, in the time-sequence of images is an important contribution.
- We have built a fully functional prototype that implements CDD on existing smart phones. We have tested the prototype in real-world conditions and demonstrated that it is possible to detect substantive and sudden disruptions without triggering false positives (such as when the ambient lighting changes). Image analysis is one of the hardest problems in terms of computational requirements. By demonstrating the feasibility of this problem with COTS (commercial, off the shelf) products we have validated the concept and its effectiveness in terms of cost and performance.
- We have developed an escalation procedure that the smart phone can activate in stand-alone mode. The general philosophy of the escalation procedure is akin to that of the alarm clock. – the clock sets off an alarm, the user can

deactivate (hit stop), gather additional information (look outside or look at the time) or schedule for re-escalation. When the CDD algorithm initially triggers, it uses a low-cost low-reliability channel such as Wi-Fi or VoIP. Then based on the feedback from the user it can de-escalate, hyper-escalate and/or provide additional information such as a still feed or a live video feed. We have developed an archival server that can push information to clients such as browsers. But the phone is self-contained and controls the entire escalation procedure in conjunction with the user by employing different modalities and transmitting varying levels of information for generating situational awareness.

II. Design Consideration & System Choices

A. Project Requirements

At the outset we set forth some basic project requirements. The mobile cell phone or smart phone must be a phone which can be used in USA, Europe (with EDGE, UMTS, HSDPA, a plus) and India. This requirement was based on the location of the authors and the intended target use of the application. We should utilize only the computational power of the cell phone for image processing. The cell phone must be highly independent (i.e. no permanent connection between cell phone and server).

As part of our project feasibility study we had to investigate which technologies (Operating systems, APIs) are required to automatically take pictures. We had to list and compare available compatible phones and standardize on one of them for the project. We had to do research on various image recognition/comparison algorithms and analyze the problem of coherence disruption detection. We had to develop, implement and test the image processing algorithms. We were required to develop the mobile application to develop the server application, and last but not least to test the complete application in real conditions

B. Basic Architecture

As per the requirements laid out at the outset, most of the time, the cell phone should not be connected to the server, because we want the cell phone to be highly independent. So, when the cell phone must trigger an alarm to the server, it has to connect to the server, using WiFi network (if available), otherwise using the GPRS/UMTS network. The server should also be able to contact the cell phone for example to change the configuration of the cell phone (start, stop, change interval time, etc). Consequently both the server and the cell phone) have to be able to initiate connections with each other.

See Fig 1 for the basic network architecture. We assume that the server is always up and connected to the Internet and the cell phone is localized in a GPRS/UMTS cell (or in a Wi-Fi network range). The cell phone (CP) and the server (S) can contact each other by connecting through a TCP socket, the IP address of the server is obtained through DNS resolution. There are two bidirectional channels, because the server must be able that are completely independent, and each of S and CP implements a client and a server. We adopt a simple authentication using a shared secret key, same as a challenge-handshake protocol (CHAP) and OpenSSL based encryption.

We use AJAX on the server to implement an interactive GUI.

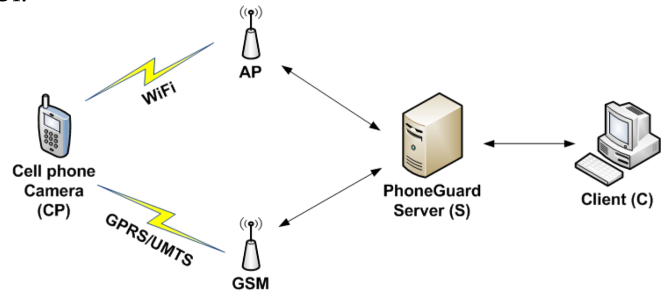

Fig 1. Basic Network Architecture.

C. CellPhone Choices

There were several constraints about the phone. The phone must have a camera, because the camera is the heart of the project. It must be usable in USA, EU and India. And it should be cheap. Furthermore, the phone should communicate via WiFi and possess a relatively powerful processor to perform image processing. It should also have extension possibilities (ROM, memory)

Our investigation indicated that the biggest differentiator was that some system allowed the phone to take pictures automatically while others required explicit manual intervention rendering them useless for this project.

Platform	**Remarks**
Windows Mobile Version 5.0 (Pocket PC or Smartphone)	Native (c++) or managed code source (C# or VB.net). The API allows to take still pictures (.jpeg but can be converted into another formats) or videos (.WMV) via the Camera Capture API
Blackberry	Java JDE or .NET compatible. The API does not permit to take pictures (only a Listener can used to trigger an application when

	a picture is manually taken)
Palm OS 5.0	In C language, the API allows to take pictures via the Cameralib library in a bitmap formats and videos in various format (QVGA, QQVGA, QCIF, CIF). No image or video conversion function in the API

Table 1. Comparison of different cell phone platforms.

Based on our comparison we shortlisted the HTC Cingular 8525 and a Palm Treo 750. We obtained a Cingular 8125 cheaply from eBay and proceeded to test it. We found that MMAPI was not implemented on it but reference code from the Windows Mobile Demo projects successfully compiled and allowed us to take pictures remotely. So the decision was made to develop the cell phone software on C# [10] on the .NET Platform with the Cingular 8125 phone.

D. Server Alternatives

The server is divided in two parts. The back-end of the server must receive the pictures and alarms from the cell phone. Therefore the software must listen for an incoming connection. To implement the remote control of the application, the server should also be able to actively open a socket to the cell phone. The front-end of the server must interact with the client through a web server and Ajax. We chose to split the server into two independent parts with the back-end being Java-based and the front-end based on an open source PHP library. The communication between the two parts is based on the OS file system and a database. The front-end of the server is client-agnostic so long as the browser is Javascript (AJAX) compliant.

E. Complete Architecture

See Fig 10 at the end of this paper for the complete network architecture.

III. COMPARATIVE IMAGE ANALYSIS

A. Overview of the Coherence Disruption Detection (CDD) Algorithm

The graphic in Fig 11 at the end of the paper shows the entire image processing chain. The following is the list of algorithms to be implemented on the phone:

- Conversion to gray scale
- Histogram equalization
- Image difference
- Low pass filter (smoothing filter)
- Enhanced 8-neighborhood weighted algorithm

One of the main tasks of the project is the image comparison. The reliability of the alarm system depends on the quality of these image comparisons.

If it is relatively easy to compare two completely different pictures, it is much more complicated to distinguish between two slightly different pictures [9]. In the digital world, a picture can be considered different from another if at least one of the pixels is different. Two pictures taken back to back are almost always different, even if the conditions are the same. However, a good image comparison method should only detect relevant errors or coherence disruptors. A coherence disruptor can be crudely described as a relatively large object appearing in the picture. Moreover, the image comparison should not be disturbed by brightness modification or other noise. Rather than present the final Coherence Disruption Detection Algorithm we present it in two stages, a basic version followed by an enhanced version, as this will enable the reader to better understand the design decisions.

B. Basic CDD Algorithm Chain

The first step is to quantify our intuitive notion of a qualitative difference and later, set an adjustable threshold to decide whether an alarm must be triggered [7]. For the analysis we consider 8-bits gray level pictures (512x512 pixels), and compare two pictures (A and B) between them. Considering the pictures as matrices, a difference picture (D) can be made by computing the difference for each pixel

$$D_{m,n} = |A_{m,n} - B_{m,n}| \text{ for all } m \varepsilon [1..M], n \varepsilon [1..N]$$

where the picture is M x N pixels large. The naive approach to quantify this difference (S, for score) would be simply to sum all the values of the difference picture D :

$$S = \sum_{m=1 \text{ to } M} \sum_{n=1 \text{ to } N} D_{m,n}$$

But the problem is that all the error pixels are not of equal importance. One aspect of the human perception of differences between two pictures is that greater the size of the difference object the greater is the perceptible difference. To increase the impact of wide area errors, the error of a given pixel is multiplied (weighted) by the number of error neighbors. The idea is to increase the impact of large objects, which are relevant differences or valid coherence disruptions. This way, the impact of small areas error will be negligible compared to wide area errors. Furthermore, a completely isolated error pixel will have no impact (because it has no direct neighbors, it is multiplied by 0). Thus, it is possible to define the partial score for each pixel:

$$S_{m,n} = D_{m,n}\left[\left(\sum_{i=m-1 \text{ to } m+1}\sum_{j=n-1 \text{ to } n+1}(1 - \delta(D_{i,j},0))\right)-1\right]$$

where δ is the Kronecker delta function and the outer -1 serves to eliminate the considered pixel itself. Then, the total score is the sum of all partial scores of the picture:

$$S = \sum_{m=1 \text{ to } M} \sum_{n=1 \text{ to } N} S_{m,n}$$

An advantage of this algorithm is that isolated (no neighbors) error pixels are ignored. The default behavior of this algorithm is that a pixel is considered as an error even if the error is very small. In reality, almost all pixels are slightly different from one picture to another, due to noise. An enhancement of this algorithm is that only pixels with an error above a specific threshold are considered errors. A solution is to replace the Kronecker delta function by the Heaviside delta function:

$$H(x; th) = 1 \text{ if } x > th \text{ and } 0 \text{ otherwise}$$

We can now consider only the pixels greater than a fixed error threshold, instead of all pixels :

$$S_{m,n} = D_{m,n}\left[\left(\sum_{i=m-1 \text{ to } m+1}\sum_{j=n-1 \text{ to } n+1}(1 - H(D_{i,j}, th))\right)-1\right]$$

Again, the total score is the sum of all pixels' partial scores:

$$S = \sum_{m=1 \text{ to } M} \sum_{n=1 \text{ to } N} S_{m,n}$$

This is the final version of the 8 neighborhood algorithm. Note that now there is a threshold parameter to the scoring process. This algorithm has been implemented using Scilab-4.0 [13] and the SIP Toolbox (Scilab Image Processing Toolbox) [14], see Box 1.

The score is not an absolute value and can be expressed as a ratio of the maximal score. The advantage of using a relative score is that it is independent of the image size. A small score refers to a small difference (score is 0 if the pictures are exactly the same, considering the error threshold).

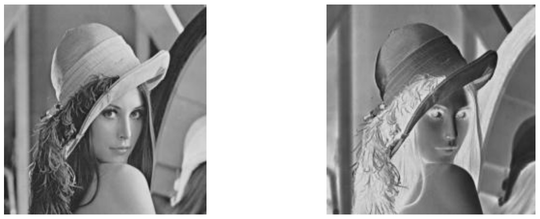
Fig 2. Lena original (score 0%) and Lena inverted (score 30.56%).

The disadvantage of the basic version of the algorithm is that it is too sensitive to brightness modifications. We therefore consider a normalization in the enhanced version.

C. Enhanced CDD Algorithm Chain with Histogram Equalization

In order to reduce the impact of brightness on the image, we perform an histogram equalization before the weighted 8-neighbors algorithm scoring process. First, we compute the histogram of the picture, defined as follows:

$$hist(k) = \sum_{m=1 \text{ to } M} \sum_{n=1 \text{ to } N} \delta(x_{m,n},k)$$

Here vector hist contains the number of appearances for each possible k value between 0 and 255. Histogram equalization serves to increase the dynamic range of the histogram. The goal is to get an uniform distribution of the intensities on the image. As the process is discrete, a completely uniform distribution is not possible in the most of the cases. The effect of a histogram equalization is a sort of normalization of the brightness intensity of the picture. After the equalization, the images comparison is less sensitive to brightness modifications. Mathematically, histogram equalization can be defined as a cumulative sum, including a correction factor :

$$Hist(k) = \left\{\sum_{i=1 \text{ to } 256} hist(k)\right\}*255/MN$$

Box 1. Scilab implementation of basic version

```
 [oi,om] = imread('lena.bmp') ; // original image
[mi,mm] = imread('lena2.bmp') ; // image to compare with
odiff = abs(oi-mi) ; // compute error image
score=0 ;
    threshold=10 ;
for i = 2:x-1,
    for j = 2:y-1,
        if odiff(i,j) > threshold then
            neighbor = 0;
            for k=i-1:i+1,
                for l=j-1:j+1,
                    if odiff(k,l) > threshold then
                        neighbor = neighbor+1 ;
                    end;
                end;
            end;
            score = score + (neighbor-1)*odiff(i,j)/255 ;// -1 :
remove central point
        end;
    end;
end ;
```

This process is also known as the accumulated normalized histogram. Now the new histogram Hist(k) can be assigned to the image, using a lookup table:

$$Deq_{m,n} = Hist(D_{m,n})$$

See Box 2 for the Scilab implementation of the histogram equalization. Now we can compare the score for image comparison with and without histogram equalization.

The great improvement is that now the images comparison is not sensitive to image contrast and brightness difference.

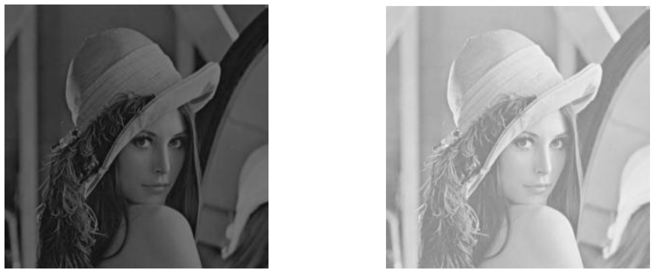

Fig 3. Lena low-brightness (score 0%) and Lena high brightness (score 0%).

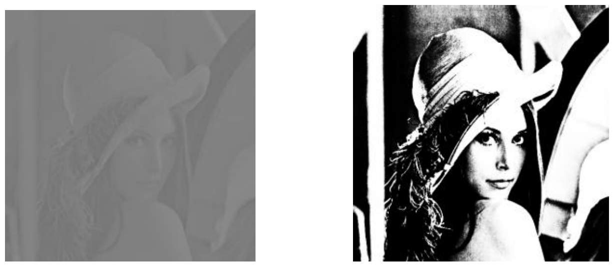

Fig 4. Lena low-contrast (score 1.5%) and Lena high-contrast (score 20.81%).

IV. Implementation

The architecture has been realized using Object Oriented concepts, enabling easy reuse. The source code is well-structured and well documented.

The image processing required a number of hacks and tweaks to obtain the requisite performance. One of the major hacks was the use of the unsafe directive [15]. Using unsafe methods, the computation time was improved by a factor 8 to 10. The disadvantage though is that a more complex syntax must be used to access the memory, using pointers. Pointers offer great flexibility but greatly complicate programming and debugging. C# .net hides most of memory management

Box2: Scilab implementation of histogram equalization

```
function [imn, hist,
mhist]=hist_equalization(im)
hist = [0:1:255] ;
for i=1:256, hist(i)=0 ;
end;
// build histogram
[x,y]=size(im) ;
for i = 1:x,
for j = 1:y,
hist(im(i,j)) = hist(im(i,j))+1 ;
end;
end;
// compute new histogram values
mhist(1) = hist(1)*255/(x*y) ;
prev = mhist(1) ;
for i = 2:256,
prev = prev+hist(i) ;
mhist(i)= int(prev*255/(x*y)) ;
end;
// modify image with LUT
for i = 1:x,
for j = 1:y,
imn(i,j) = mhist(im(i,j)) ;
end;
end;
endfunction [oi,om] = imread('lena.bmp') ; //
original image
[mi,mm] = imread('lena2.bmp') ; // image to
compare with
odiff = abs(oi-mi) ; // compute error image
score=0 ;
    threshold=10 ;
for i = 2:x-1,
   for j = 2:y-1,
      if odiff(i,j) > threshold then
         neighbor = 0;
         for k=i-1:i+1,
            for l=j-1:j+1,
               if odiff(k,l) > threshold then
                  neighbor = neighbor+1 ;
               end;
            end;
         end;
         score = score + (neighbor-
1)*odiff(i,j)/255 ;// -1 : remove central point
      end;
   end;
end ;
```

A. CellPhone Development

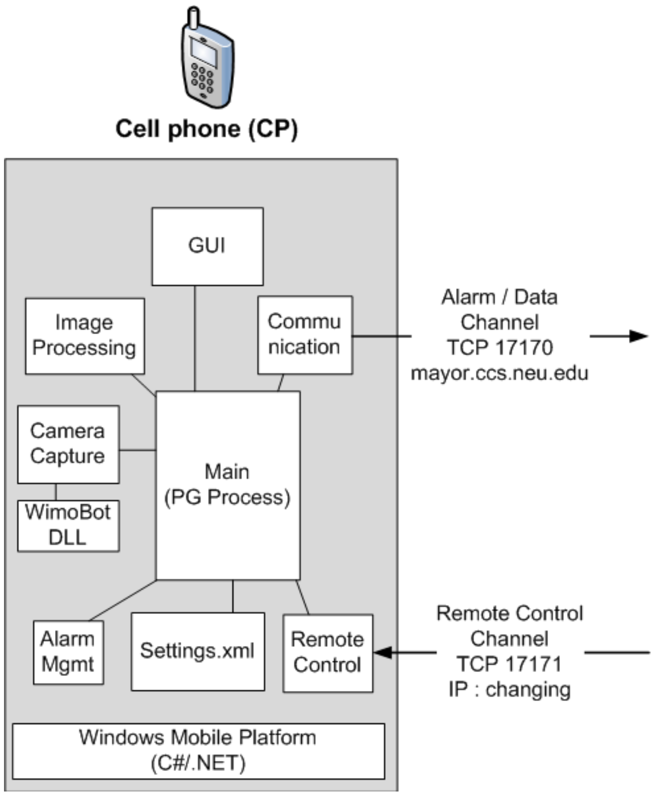
Cell phone (CP)
GUI
Image Processing
Commu nication
Camera Capture
WimoBot DLL
Main (PG Process)
Alarm Mgmt
Settings.xml
Remote Control
Windows Mobile Platform (C#/.NET)
Alarm / Data Channel TCP 17170 mayor.ccs.neu.edu
Remote Control Channel TCP 17171 IP : changing

Fig 5. Cell phone architecture

which makes it much easier for the developer - thanks to the Garbage Collector and the use of references. But to make the language powerful enough in some cases in which we need direct access to the memory, unsafe code was invented. As a result of the use of this hack our project must be compiled with /unsafe option to allow unsafe methods

B. Server Development

The server is the central part of the system. The main tasks of the server are:

- Handle the incoming alarms/pictures sent by the cell phone
- Push the information (alarms/pictures) to the client
- Send the remote control messages to the cell phone

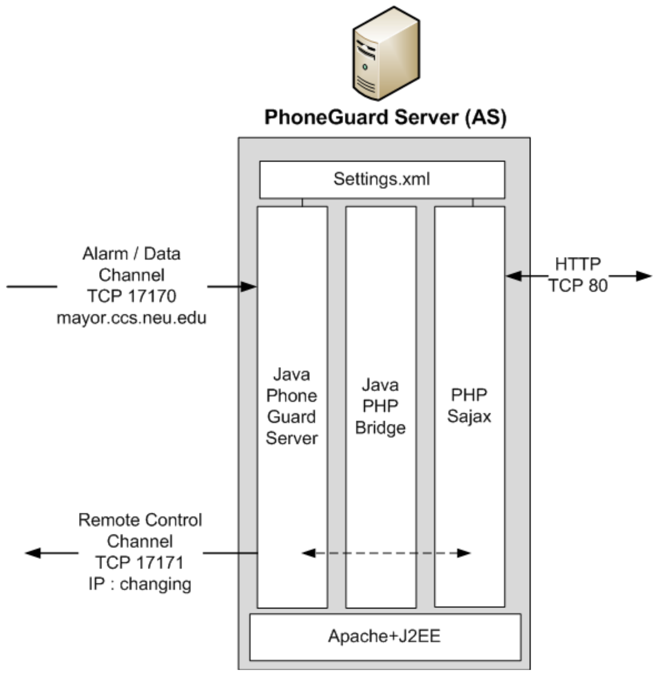

Fig 6 Server architecture.

On the cell phone side of the server, a TCP server has been developed in Java to listen to incoming connections. The TCP server also has the functionality to actively open connections to send commands. On the client side of the server, an interesting and powerful library called Sajax [16] was utilized to easily add Ajax features. In the middle, the PHP/Java bridge connects both parts of the server.

C. Alarm Escalation Procedure

The alarm consists of an alarm source, a server and a client monitor. When an alarm occurs, the information about the alarm is uploaded to the server and pushed to the client.

The three basic situations are described in the following figures.

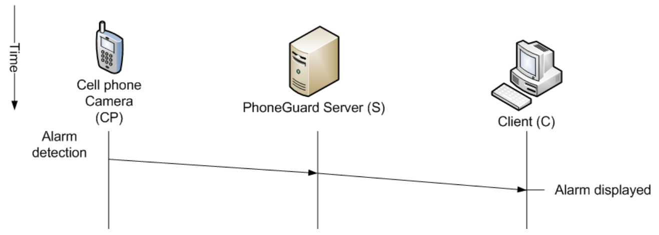

Fig 7 Alarm triggering without escalation.

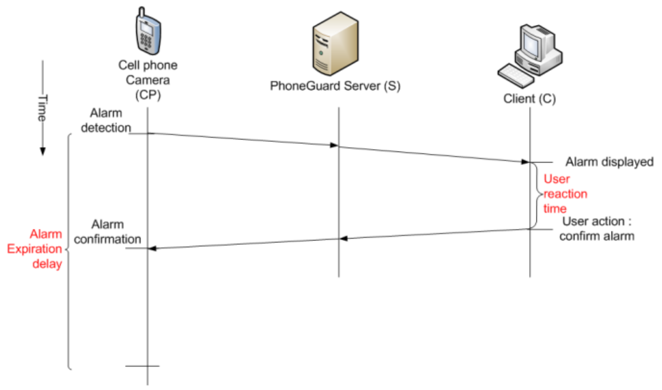

Fig 8 Alarm escalation with confirmation.

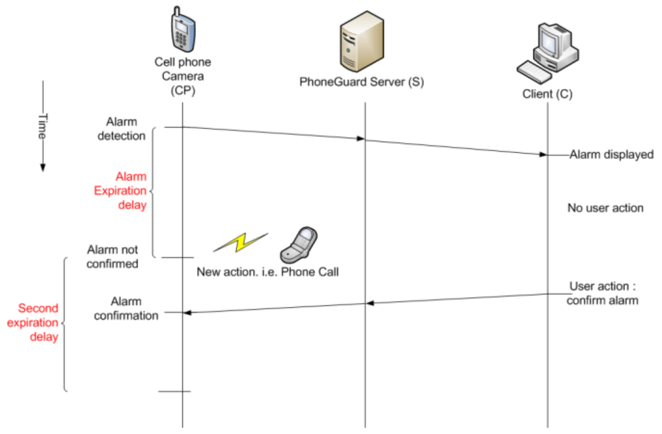

Fig 9 Alarm escalation without confirmation.

The following are the different escalation procedures implemented [8]:

- Phonecall using GSM network
- Phonecall using VoIP network
- Email
- SMS through GSM network
- SMS through a 3rd party
- MMS through GPRS/3G network
- Video call using GSM /3G network
- Delayed video clips.

D. *Testing*

Testing was done using a diverse yet representative set of pictures. The main finding from testing was that controlling the percentage of false alarms is critical for a useful system. In fact, the idea of histogram equalization arose from the first round of testing where we discovered a false alarm rate as high as 10%. The deployment of histogram equalization subsequently brought the false alarm rate below 1%.

A large proportion of the testing time was spent tuning the various parameters particularly the threshold. We call the feasible region the k-zone.

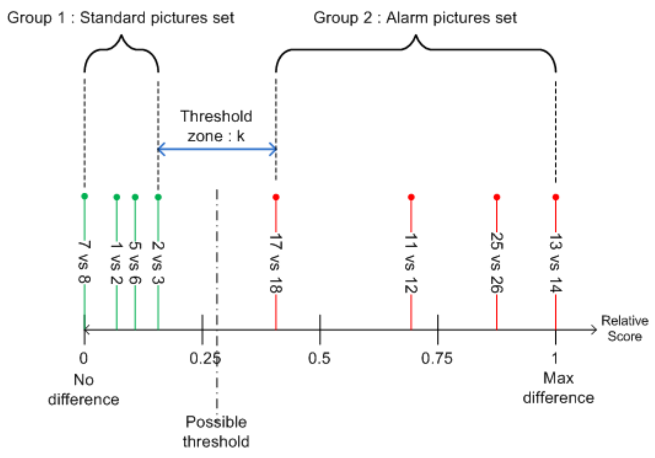

Fig 10 k-zone

V. STRENGTHS AND WEAKNESSES

Our current design has a number of strengths and weaknesses. We briefly elaborate on them.

The main strengths of our PhoneGuard system are

- It does not require any special infrastructure like a wireless sensor network. It uses existing cellular infrastructure which has widespread coverage even in places such as India and Africa.
- It can be deployed even by one person who wishes to be notified of an impending disaster or crisis.
- It is easily extensible to other kinds of sensing modalities, e.g. use of audio for sound (e.g. water leaks), accelerometer for motion (e.g. earthquakes) etc.
- It is easy to setup and inexpensive to deploy requiring just a smart phone with a cellular plan.

However, the PhoneGuard system also has its drawbacks. Some of the major drawbacks are:

- It requires the presence of cellular infrastructure
- Mobile devices have limited battery life and need to be recharged periodically. Alternately, in remote monitoring scenarios the mobile device must be tethered to a power source.
- Smart phone CPUs are limited in their capabilities and cannot do compute-intensive tasks such as sophisticated image processing. Of course, the computational capabilities of smart phones are improving all the time.
- If the smart phones are used in ad hoc fashion for disaster notification without being secured properly then they may get knocked over or shifted around and thus may not function properly.

- Smart phones are also vulnerable to Denial of Service attacks such as by repeated phone calls or by SMS/emails that may contain viruses, worms and other Trojans. However security too is improving all the time and it is possible to get special plans (e.g. chaperone plans for kids) that restrict phones only to communicating with a pre-specified set of numbers.

VI. Conclusion

We conceived of a smart app to aid with disaster prevention and created a fully working, comprehensively tested prototype. A novel feature of our prototype is the fact that we are able to carry out the entire image processing chain on the mobile device.

In future work we propose to expand the set of sensing modalities to include audio and motion. It is our belief that, with the growing ubiquity of cellular infrastructure and smart phones, such apps will be the future of disaster prevention and management.

Acknowledgment

We thank Chancellor Venkat Rangan for informing us of the conference on wireless technologies for humanitarian relief.

References

[1] W. Dargie, and C. Poellabauer, "Fundamentals of wireless sensor networks: theory and practice", pp. 168–183, 191–192, John Wiley and Sons, 2010.

[2] K. Sohraby, D. Minoli, and T. Znati, "Wireless sensor networks: technology, protocols, and applications," pp. 203–209, John Wiley and Sons, 2007

[3] M. Mamun, Y. Koi, N. Nakaya, and G. Chakraborty, "A novel integrated wireless sensor network architecture for disaster prevention," International J. on Smart Sensing and Intelligent Systems, 2(2), 2009.

[4] A. Khan, and L. Jenkins, "Undersea wireless sensor network for ocean pollution prevention", in Proc. of *Communication Systems Software and Middleware and Workshops, (COMSWARE 2008)*, IEEE, pp. 2-8.

[5] Pet Remote Monitoring through Smart Phone Application, www.asmag.com/showpost/11810.aspx, 2011.

[6] D. Ryu, H. Na, and S. Nam, "Implementation of a USN-based disaster prevention system in Korea," International Journal of Computers, 3(1) 2009.

[7] S. Egger, and M. Schrag, "Mobile Pattern Recognition," EIA-FR Semester project 2006.

[8] O. Caille, "Seamless and secured push service on heterogenous networks," EIA-FR Diploma work 2005.

[9] G. A. Baxes, Digital Image Processing, Principles and applications, John Wiley & Sons, Inc., 1994

[10] C. Petzold, Programming in the key of C#, Microsoft Press, 2003.

[11] Wireless communications in India, Wikipedia, http://en.wikipedia.org/wiki/Communications_in_India#Mobile_telephones.

[12] Bhedaghat, http://en.wikipedia.org/wiki/Bhedaghat.

[13] Scilab, http://www.scilab.org

[14] SIP (Scilab Image Processing) Toolbox, http://siptoolbox.sourceforge.net/ .

[15] C# unsafe directive, http://www.codersource.net/csharp_unsafe_code.html .

[16] Sajax, http://www.modernmethod.com/sajax ..

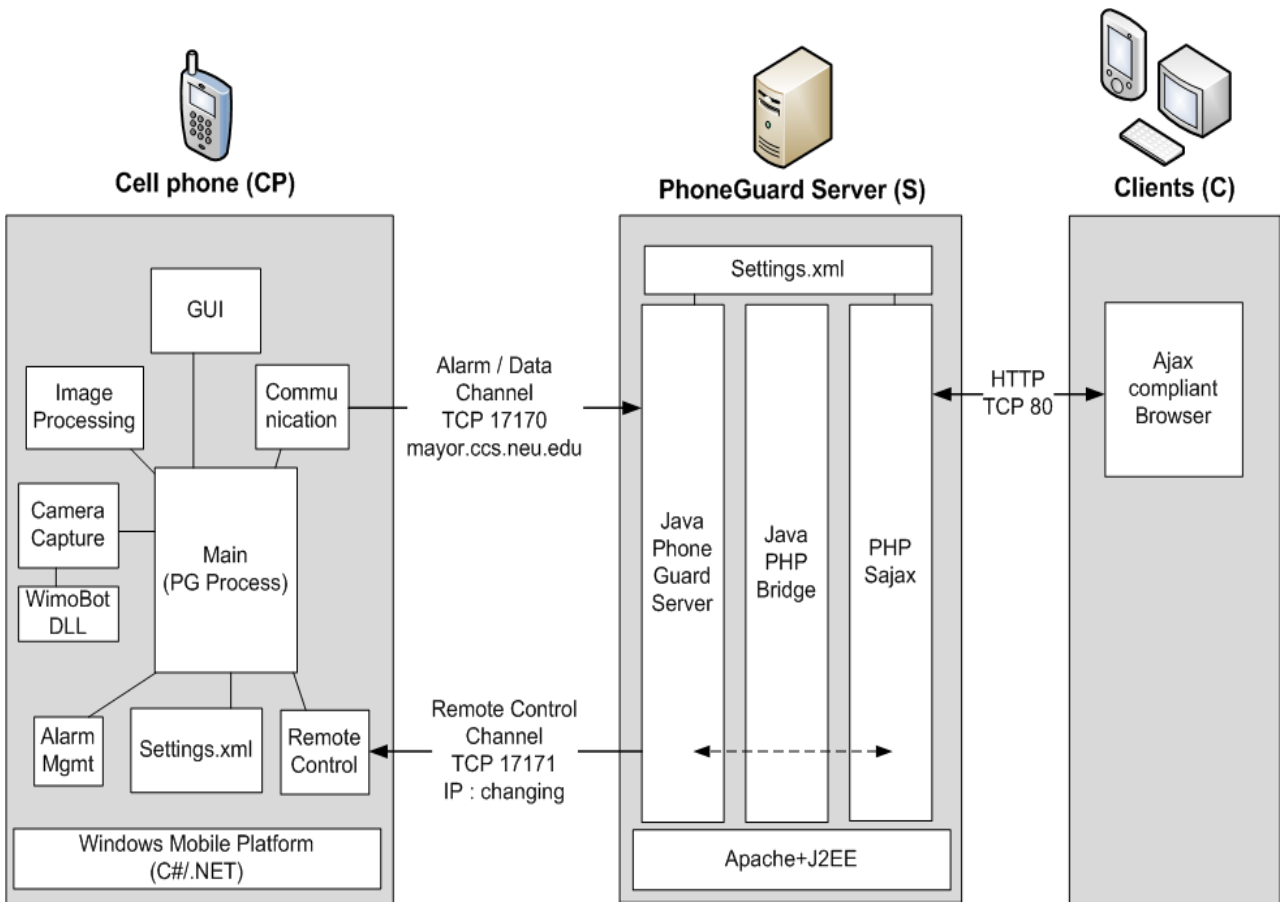

Figure 10. Complete network architecture.

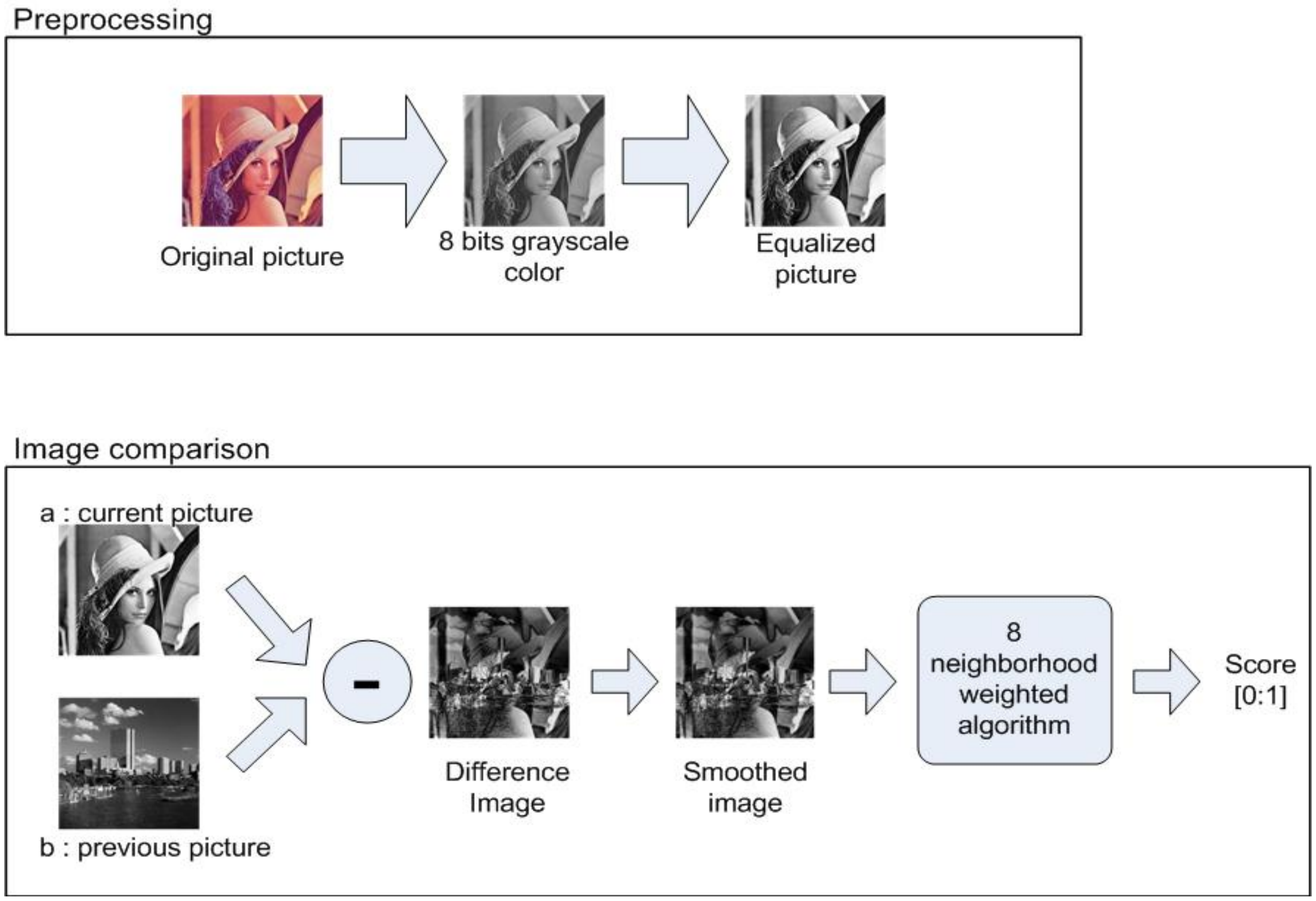

Figure 11. Complete image processing chain..