Department of Electrical Engineering

Presents

**CamCare**



*An image processing design to help the elderly feel safe at home.*

**Senior Design Final Report**

**Spring 2010- Fall 2011**

Project Advisor: *Muhammad Ali.*

Team Members:

*Rigers Qeraj*

*Jorge Correa*

*Luis Santos*

*Wuji Chen*

**Table of Contents**

**Preamble**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Chapter 1: Image Processing**

Introduction\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Platform\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Working Principles\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Coding\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Chapter 2: SMS Alerts**

Introduction\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Platform\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Working Principles\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Coding\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Chapter 3: Testing and Implementation**

**Preamble**

Our capstone project consists of designing a system that detects when an elderly person falls. The major problem when accidents amongst elderly happen is that help doesn’t reach them fast enough. Our project uses image processing to detect when a fall has happened and then automatically send a SMS to a relative or guardian of the elderly.

**Chapter 1: Image Processing**

*Introduction*

There are many ways in which people fall so in order to make our software as accurate as possible we had to devise a system that will work for all occasions. One possible method would have been to stage as many falls as possible gather the data and have them stored in memory. This however didn’t prove to be very productive as there are many different methods in which a person can fall. Also it isn’t very efficient, as the motion would have to be compared with all the data stored in memory, which wastes time and could slow down the system considerably. Another issue that we had to consider was differentiating a person falling from someone voluntarily bending. These issues create choices for as to what constitutes a fall and what doesn’t so creating a reasonable threshold is vital. The major problem for us is not to produce false positives but false negatives. Essentially it is better to get a false alarm rather than no alarm when a fall does occur; for this reason we opted for a more universal approach and decided to analyze speed and direction instead.

*Platform*

In order to make the image processing as versatile as possible we had to choose a platform that is well developed. We decided to go with Open Computer Vision (OpenCV). OpenCV was developed at first by Intel but later was released to the public as an open source project. We chose OpenCV because it’s open source, very well documented, supported by all 3 major OSs (Mac OS X, Windows, Linux), and can be implemented using both C/C++ and Python.

*Working Principles*

As we mentioned before we decided to measure speed and direction in order to achieve our goal of detecting an accident from a voluntary motion. Since we know the frames per second (FPS) we can calculate the relative speed of a moving object by calculating the distance of the object traveled in the image and dividing the result by the FPS. Also we can analyze the motion of an object and tell the angle of direction relative to the horizontal. After finding the speed and direction of motion we created a threshold level that tells us what is a fall what isn’t. This, of course, can be changed easily to allow for a precise calibration of the individual that is being analyzed.

In order to analyze the image we got rid of the background, so that we can view only the moving objects. However, since people could have pets and analyzing their motion as well could provide errors, we implemented a people detection algorithm that makes it possible to analyze only the motion of humans.

*Coding*

**Chapter 2: SMS Alerts**

*Introduction*

After a fall has been detected, the next step is to alert the caregiver. This will be done by sending an automatic SMS once the fall has been properly detected. The caregiver will then decide what the proper action to take is. In order to achieve this, the program first needs to make sure that a fall has occurred. When a positive fall has been detected the program generates an XML file. This XML file contains the information whether a fall occurred or not.

*Platform*

Google-voice among many other things allows us to send SMS. We first had to create a Google-voice account for this to work. The program has to check constantly if there is a file in a previously specified location on the computer. As described before, the file will be an XML. The loop will constantly be checking until it finds the file. Then it proceeds to parse the file with the information that it contains. Next, after reading and printing the contents of the file, the program will connect to Google Voice which provides us the origin number at the moment of creating the account.

*Working Principles*

This section proceeds to explain how the message routine works. As described before, we have to install the libraries from the Google Voice and the DOM Parser. This needs to be done in order to send the message and read the file respectively.

Since the program keeps constantly running checking the file, we proceed to declare a while loop so the program is always running. The code below does the following:

boolean condition = true;

while(condition){

File fXmlFile = new File("C:\\testfile.xml");

boolean exists\_2 = fXmlFile.exists();

The first line declares a Boolean variable named “ condition ” and initializes to true. This will be done to run an infinite while loop which the program needs to makes sure that it is constantly running. The third line creates an object reference to the Xml file and stores it in the variable “fXmlFile”. The fourth line declares a boolean variable named “ exists\_2 ” and initialize with the reference to the file “ testfile.xml”, which should be in the hard drive. This means that at this point this condition is true.

Next, after we run the loop, we used an if-else loop to check if the file is in the specified location or not:

if (!exists\_2)

{

System.out.println(" File is not there. Keep checking:" + exists\_2);

Thread.sleep(5000); // wait 5 seconds before checking again

}

The lines of code above do the following: the if condition checks is the file is not there and reads it, and it will keep checking every 5000 miliseconds or 5 seconds until it reads if the file again. The Thread.sleep() method stops the program for the specified amount of time before restarting again.

As long as the file is not there or the answer is no, the code above will be executed. However, if the file is there or it says yes, then the message routine will be executed.

private static String getTagValue(String sTag, Element eElement) {

NodeList nlList = eElement.getElementsByTagName(sTag).item(0).getChildNodes();

Node nValue = (Node) nlList.item(0);

return nValue.getNodeValue();

}

}

The lines of code above do the following:

It creates a static method of type String and names it “getTagValue”. This will be checking the file line by line and getting every object and returning each object everything this method is called. The, the program proceeds to read every line of the file and if it say yes, it will invoke the getTagValue method. Once the content is read, it will proceed to compare if it is equal to yes. If that is the case, then the message routine will execute.

First, we have to create a voice object and we do it as follows:

Voice v1 = new Voice(userName,pass);

Next, it calls the Google Voice sendSMS method and assigns to the object reference. As it can be seen below, it will use two parameters, the phone number where the message will be sent and the message content.

v1.sendSMS("1234567890", "Test message");

Finally, the message will be sent, the Xml will be closed and the program will constantly be running checking for a yes or no. According to the response, it will perform the appropriate routine.

*Coding*