NovaToast SmartVision Project Requirements

Jacob Anderson

William Chen

Christopher Kim

Jonathan Simozar

Brian Wan

Revision History

v1.0: Initial creation of the document and first draft.
v1.1 (2/9): Added video stream and Flatfield Control to Use Cases list
v1.4 (2/13): Added Toggle between Email and Text to User Stories, Average Temperature of Frame to Use Cases
v1.5 (2/19): Added more use cases and user stories, added tab structure to use case
v1.8 (2/24): Added more user stories (set only temperature field, set only fall field, load previous settings, toggle only alert on temperature or fall, test alert system), updated System Architecture Overview, Glossary of Terms, and Appendices
v2.0 (2/26): Finished UML diagram and added, added few more use cases, general proofreading

Introduction

Smart Vision is about making a “smart home” using FLIR’s mobile thermal sensor, the Lepton. The Lepton will be used to monitor an area in a house, and send notifications based on certain events, such as a person falling, frost on a window, or if an object is blocking the camera. The interface will be controlled by an external web application.

With each feature, there are several challenges we must consider. If we implement fall detection, we need to be able to differentiate between an actual fall or something like a pet sleeping on the floor. In the case of the temperature monitor we need to distinguish when an area will remain at a certain temperature, such as a fire starting in your house and ignore something like a lit candle.

Aside from the feature challenges, there several general issues we must face. One in particular be detecting if the camera is blocked. If the camera is blocked, no events can be detected. Another issue is if the camera shuts off unexpectedly. A similar problem is if the network connection is broken. These two cases would ultimately result in Smart Vision being unusable. These are several of many possible issues that we may face and we must decide how to handle and account for each appropriately.

The problems that can occur in homes have solutions today in the form of many different products, but not one single system. Currently, there are no products that automate the home as Smart Vision does, but many are certain to follow. Smart Vision will be an all-in-one system that can serve multiple purposes in many applications.

Glossary of Terms

- **Framework**: The underlying structure in a system.
- **GPIO**: General purpose input-output; a bus system for a wide range of applications.
- **GUI**: Graphical User Interface. An interface with which the user will interact made of graphical elements.
- **HOG**: Histogram of Oriented Gradients. This is a object detection algorithm that uses gradient orientation concentration to differentiate features.
- **I2C**: A 2-way serial computer bus used for connecting low-speed devices to larger systems.
- **Lepton**: A mobile thermal imaging sensor developed by FLIR.
- **OpenCV**: An efficient image processing library available for many languages. Used for real-time applications.
- **Radiometry**: Techniques for measuring electromagnetic radiation, or more specifically, the distribution of radiation power in space.
- **Raspberry Pi**: A low cost, miniature computer system offering an ARM-based processor.
- **SPI**: Serial Peripheral Interface; allows synchronous serial communication between devices.

System Architecture Overview

The initial setup will be with a Raspberry Pi Model B connected by GPIO to the Lepton module. The Raspberry Pi is a small computer that contains a stripped-down Linux distribution. It has a Broadcom system-on-a-chip with a 700MHz ARM processor, 512 MB of shared RAM, and a 100 Mb ethernet port. There is a concern that the Pi will not be powerful enough to match the software specifications in which case, either using the newer Raspberry Pi 2 or a more powerful Windows machine will be used as a backup. By the end of the project, ideally, the Lepton and the Pi will be housed in a custom 3D-printed casing.

The Lepton is a longwave infrared imaging sensor. The particular model used has no shutter, so it works from calibration, meaning, once calibrated, it will be able to detect temperatures colder or hotter than the object which it was calibrated to. With the potential addition of a shutter, it will be able to be accurately calibrated every time. RBFO calibration values will be read from the Lepton and used in various radiometry algorithms to obtain an absolute temperature value. The Lepton sends the image as a stream of bits over a serial connection which is then pieced together in the Pi.

The interface between the camera and processor uses SPI and I2C. SPI feeds data from the Lepton into the Pi, while I2C is what the Pi uses to issue commands to the Lepton.

The Pi will be controlled by a web interface built with a mixture of HTML/CSS, Javascript, Java, and the Django framework. This application will first scan the network for the Pi, then connect to it. Once the connection is made, there will be a web-based GUI that the user can open and receive input from the Pi and issue commands.

The web application is all on a single web page, and is broken up into four tabs. Using Bootstrap, the web page uses a grid system, so the content reorganizes themselves to fit in the browser as the size of the window shrinks or expands.

The first tab is a ‘Welcome’ screen, which serves to greet the user. On the tab, there are three large buttons which, when clicked, will take the user to the other tabs. Alternatively, the user can click the tabs on the top of the web page.

The second tab is the ‘Camera’ tab. It has a container which displays the video stream from our Lepton camera. The container displays a motion jpeg frame, and as the Lepton rewrites the frame to the same file that the container is accessing, the container will refresh every one second to simulate a video. The time interval between refreshes can easily be changed in terms of milliseconds. Also on the camera tab is a ‘Flatfield’ button, which will recalibrate the Lepton when clicked.

The third tab is the ‘Alerts’ tab. This is where the user gives the Lepton parameters on which it will send an alert to the user. For now, there are two alerts: temperature and fall detection. For the temperature alert, the user inputs a temperature (in Fahrenheit). If the Lepton detects that a certain area in its view hits that temperature, it will send an alert to the user. This field can be toggled on or off, so that if the user does not want to receive alerts on the temperature, the Lepton will not send alert for it. The input field will be disabled if the toggle is set to ‘OFF’. There is also a submit button so that only the temperature will be updated. For the fall detection, the user inputs a number of seconds. If the Lepton detects a fall, and the subject that has fallen hasn’t gotten up after the number of seconds the user gives, then the Lepton will send an alert. Like the temperature option, there are toggle buttons and a submit button. Both input fields are locked to only accept numeric digits. Lastly, there is a ‘Submit All’ button, so that if the user updates both fields, the user doesn’t need to click two buttons.

The last tab is the ‘Settings’ tab. In this tab, the user can turn off all alerts, choose the type of alert, provide contact information, and send an alert to test the system. The first option allows the user to completely disable the alert system if the user chooses. If the toggle is set to ‘OFF’, everything on this tab is disabled. The second option is another toggle that lets the user choose if the alerts are sent as a text or as an email. The third option is an input field for the user to input contact information. The properties of the input field changes depending on what is selected in the toggle in the previous option. If ‘TEXT’ is selected, the label will read ‘Phone:’ and the input field will only accept numeric digits, up to ten digits. If ‘EMAIL’ is selected, the label will read ‘Email:’ and the input field will accept all kinds of characters. Lastly, there is a ‘Submit All’ button to send the settings to the Lepton, and a button to test the alert system.

The Pi will be running OpenCV, an extensive image processing library. This will be written in C/C++ for maximum performance. OpenCV will allow the Pi to take the image provided by the Lepton and analyze it for regions of interest.

The data from the Lepton is read in as a stream of 4,800 14-bit values, making an 80x60 image. Each value represents a 12-bit color with a 2-bit pad. They are then converted to qImages by the main “Lepton Engine” that will manage the buffer, check for events, and call the responses if an event is detected. Then, the qImage will be converted into a Mat which is the datatype that OpenCV uses to process images. Each event will be its own class and will run independently as threads. As of now, there are three planned classes: Temperature Region of Interest, Fall Detection, and People Detection. It is still unsure whether or not the latter two will be successfully implemented.

The Temperature Region of Interest will take in a single Mat (the most recent image from the buffer) and run an OpenCV function to find the minimum and maximum values and their locations either within a bounding box, or within the entire image. This data will be passed back into the Lepton Engine and an appropriate action will be taken.

The Fall Detection class, tentatively, will use the HOGDescriptor class in OpenCV. There are multiple existing algorithms that use HOG to detect a fall. However, there may be a problem, because existing algorithms are made for visual, not thermal, cameras. This class will read in an entire section of the buffer, and compare the frames to look for a fall. Then it will output a boolean to the Lepton Engine if a fall is detected, but only if there is a person detected in the frame which will use the People Detection class.

People Detection will work in a similar fashion, but again, there may be problems adapting the algorithms for a thermal camera. It will take in a single image and run a people detection algorithm. It will output the result either to any other class that needs it or the main lepton Engine.

A variant of IF-This-Then-That (IFTTT) will be used to communicate with phones and email for the event alerts.

Requirements

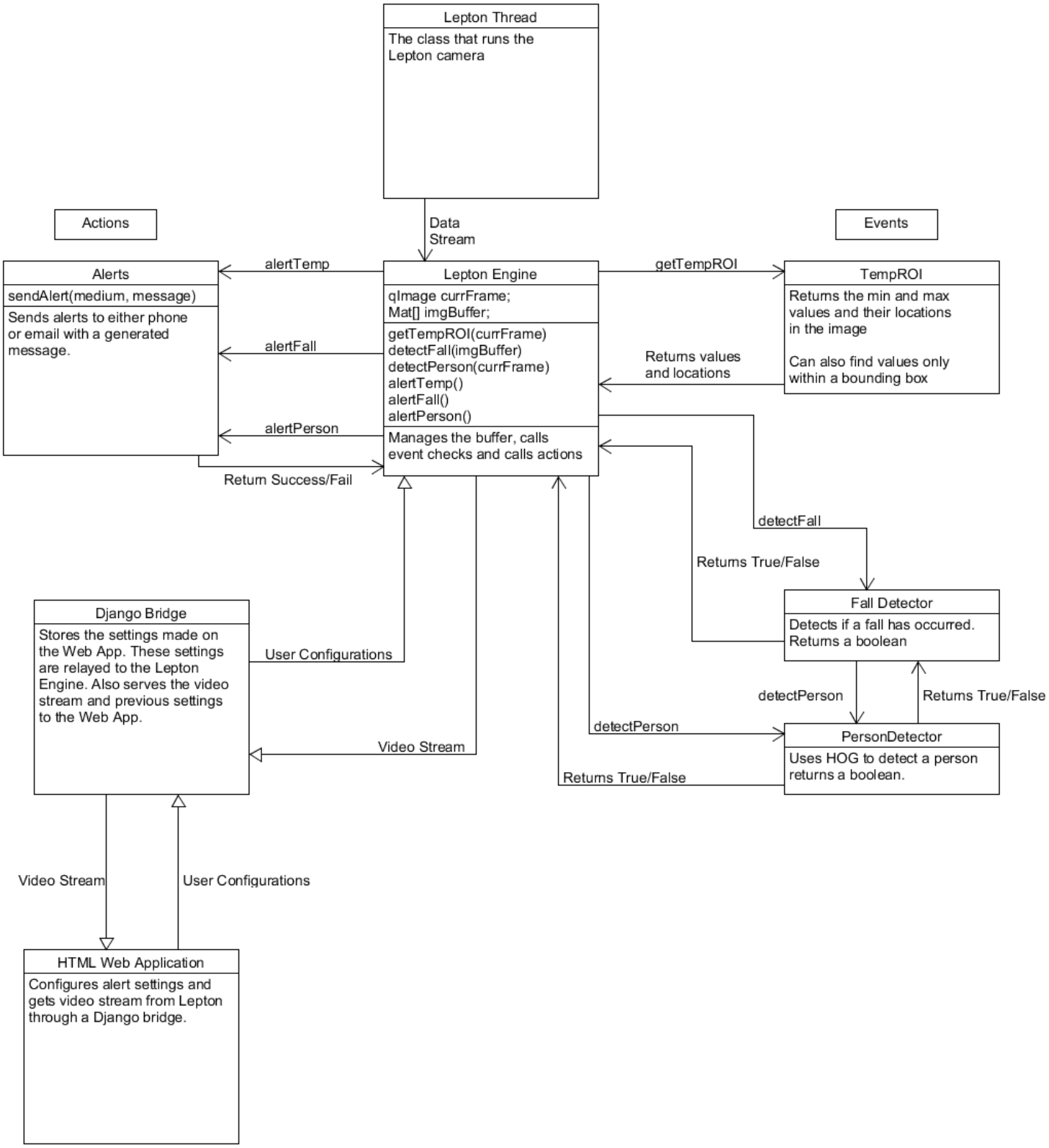
Use Cases:

1. Monitor specific area for temperature and raise a flag at certain temperatures
2. Raise a flag when someone has fallen and hasn't stood up after certain amount of time
3. Detect a human figure in the frame
4. Video Stream on web app
5. Flatfield Control on web app to re-calibrate camera
6. Send a notification if camera is off
7. Send an alert if something is blocking the camera for extended period of time
8. Detect and report the average temperature in the frame
9. Return the absolute temperature of a certain point
10. Web Application organized in tabs
11. Send a "test" alert on demand
12. Use .ini file to store configurations
13. Lepton access .ini file to set configurations

User Stories:

1. Be able to configure Smart Vision from anywhere with internet access.
2. Add user authentication for configuration through web app
3. Set phone number for alert.
4. Set email address for alert
5. Select a certain spot to monitor
6. Set certain temperature(s) to send alert
7. Set temperature units
8. Set how long a person is on the floor before sending alert
9. Toggle receiving any notifications
10. Toggle between email or text message
11. Toggle receiving notifications only for temperature
12. Toggle receiving notifications only for fall detection
13. Being able to set only the temperature field
14. Being able to set only the fall field
15. Web Application loads previous settings
16. Be able to send a sample alert to test
17. Configure low-level parameter values for the Lepton Engine

System Models



Appendices

- Hardware
  - Lepton
  - Raspberry Pi
- Software
  - CUTE, JUnit, QUnit (unit testing frameworks)
  - GitHub, SourceTree
  - Trello
  - Eclipse, IntelliJ, WebStorm, Sublime, PyCharm (text editors and IDEs)
  - Some variant of IFTTT
  - TightVNC
- Programming Languages, Libraries, and APIs
  - SPI, I2C (communication protocols)
  - OpenCV
  - Django
  - HyperText Markup Language (HTML)
  - Cascading Style Sheets (CSS)
  - Javascript
  - Bootstrap (v3.3.2)
  - jQuery (latest minified version)
  - Java
  - C/C++