University of Arkansas – CSCE Department
Capstone I – Final Report – Spring 2019

Vigilant

Haven Brown, Austin Kreulach, Ian Moncur, Lane Trobee, Zane Turner

Abstract

Many people with disabilities or health conditions(such as being prone to Grand Mal seizures) are at some risk to collapse suddenly. Additionally, often times slip and fall injuries can go unnoticed for a long period of time, as such, our goal is to use computer vision to recognize slip and fall. This would have the potential to reduce response times in key situations as well as reducing liability in slip and fall cases.

1.0 Problem

In public spaces, whether a public park, grocery store, or mall, oftentimes an injury (such as a seizure or heart attack) can go unnoticed for key minutes that are critical in making sure that treatment is delivered as soon as possible. Additionally, oftentimes these spaces are already monitored by security cameras that computer vision could use to recognize a potential injury.

Additionally often times in slip and fall cases where liability plays a role getting footage of a slip and fall can be complicated due to the amount of footage generated by an entity (i.e. Corporations or CCTV) and footage relevant to such a case may be deleted long before it is asked for, or deemed necessary in a relevant court case. If the slip and fall could be recognized, footage could be put into a longer term storage in the event it's deemed relevant to a case.

2.0 Objective

The objective of this project is to use computer vision to recognize a fall.

3.0 Background

3.1 Key Concepts

This project is fundamentally about applying computer vision to a particular use case. Computer vision is a topic in computer science relating to the application of various graphics algorithms to the task of interpreting images or video in various ways. The goal is to first determine if a human

is in the frame using edge detection, second determine the pose and thus, status of the human using motion history images.
Edge detection is a basic idea in computer vision that discovers boundaries between objects, edges, by detecting sudden changes in the gradient of the image. That is, it detects edges by looking for abrupt changes in the color between two pixels. This technique is essential in almost all computer vision applications that aim to detect particular shapes in the source image. In our case, we want to combine edge detection with an empirically generated template for what sort of edges imply 'human.'
The next key technology in the project is motion history images. This is an idea in computer vision wherein a history of an object's motion is stored in a grayscale image that is useful in comparing to current behavior in order to make a more information guess as to future behavior. This is useful in this project because if the human shape deviates from the recorded motion history image and the program detects that the human has also entered either the lying down or crawling state, this is a likely indicator of a fall.
A final key technology in Vigilant will be OpenCV, which is an open source computer vision library for C++ used in many computer vision projects. Vigilant may or may not make use of this system depending on whether the project is developed in C++ or another language. However it is a very common, easy to use, and highly popular computer vision solution.

3.2 Related Work

In 2007, a group of researchers at various universities in Montreal, Canada developed an algorithm with a very similar goal to the one proposed in this project [1]. Their system first detects abrupt motion and then detects if the final shape is in a common position for sitting or laying down or if the final shape is an appropriate shape for those activities. Otherwise their system suspects a fall. Their system did suffer from false positives when the user would aggressively sit down, and their system also used a very low frame rate, a single camera, and low resolution images. One possible solution is to define inactivity zones on chairs and other objects, but this could result in falls caused by a slip while sitting down to be missed by the program. The team will examine these possibilities.
Another group, also in 2007, of scholars working at various universities in Italy built a multi-camera computer vision model for fall detection [2]. This team developed a system for transferring information about the perceived action of a person between cameras in order to better inform the surveillance as the person passes between rooms. This was intended to prevent a person collapsing while at an awkward angle between cameras such that neither individually had enough information to suspect a problem. This system is more advanced than the one proposed in the original paper, presuming that the camera is able to detect falls and instead focusing on the infrastructure around the detected fall. The paper proposes that a detected fall should result in a text(SMS) being sent to an operator along with a short transcoded recording of the fall by all relevant cameras. The paper also spends a good deal of time focusing on how a multi-camera system can solve the basic problems in identifying a person in a potentially 'busy' room where their edges might be blocked by various bits of household clutter. The paper uses a probabilistic model based on prior readings of the size and

shape of the person living in the household, so that if a head is detected the system can guess where the rest of the body is. This idea is quite liable to failure, especially when switching cameras. Another proposed solution to this problem offers much more promise. Yue et al. [3] showed a system of calibrating occluded objects using a separate camera and two synchronized tracks mapped from one to the other by a transformation matrix. This method allows a multi-camera system to coordinate both cameras' videos such that both are aware of the location in their field of view even if only one can truly see the object. Vigilant may incorporate this idea in the final product.

4.0 Design

4.1 Requirements and/or Use Cases and/or Design Goals

Requirements of the product:

1. Distinguish humans from non-human items
2. Determine if a human has fallen and is having difficulties getting up
3. Capture images/video of the incident and relay this to staff of medical assistants

Use cases of the product:

1. Hospitals
    a. Expediting the speed of response to patients that have fallen
2. Public malls, public parks, etc.
    a. Provide assistance to people that might get isolated in an area and are unable to contact others for assistance
3. Private use
    a. Be able to identify important footage for slip and fall cases
    b. Over time be able to identify risk areas

4.2 Architecture

Architecture

Vigilant is intended to be a computer vision tool for analysis of live video feeds with minimal computational resources. It is capable of detecting people in the frame and interpreting their posture to determine if they have fallen. The system is lightweight, easily interpreted, and fast. It can be easily integrated into any source of video data to quickly find accurate and efficient information on the fall status of any people in the video.

This was one of our most important goals to ensure the software is usable in as many contexts as possible with minimal required modification or investiture in computing power. If our program can run on a raspberry pi with a webcamera then it should be able to run on a decade-old CCTV system in the local mall. This is essential to fulfilling our primary goal of helping people recover and survive dangerous falls.

Technologies used

The backbone of the algorithm is the open source C++ computer vision library called OpenCV, which is an industry standard in computer vision. Our code is almost entirely based on this library and uses it to detect people in the video feed and to draw a bounding box around their shape. This is critical to the remainder of the program.

As far as theory goes, the project is built on object detection and classification algorithms centering on edge detection and shape recognition. In essence the library, OpenCV, detect large values in the hue gradient of each image in the video and uses those to estimate edges between objects, and then feeds the shape of those edge outlines into a trained neural network to identify human shapes. These estimated human blobs are then returned to our program for analysis of whether that shape indicates a fall or normal behavior. This approach is common across many computer vision applications and is very well-studied.

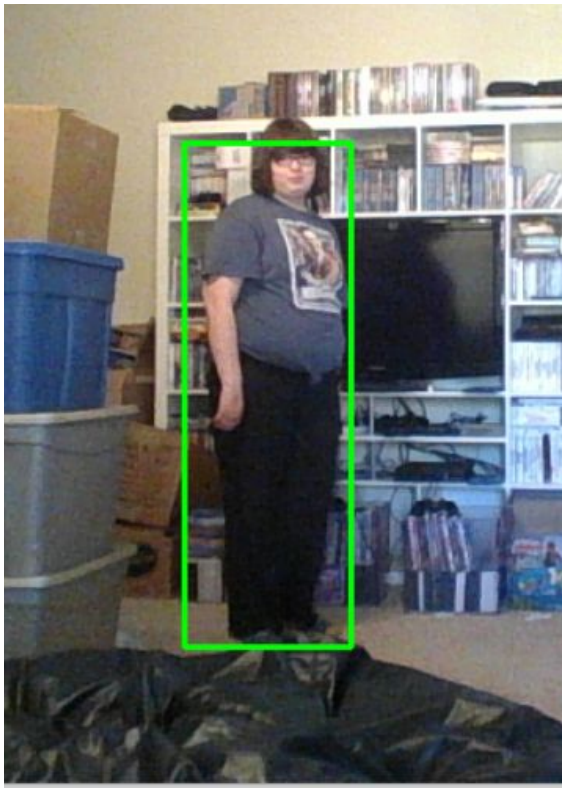
This project is applicable to a wide range of hardware. We have tested this program with the built-in cameras of our laptops in order to validate the program with various frame rates and resolutions (as seen in the implementation section)

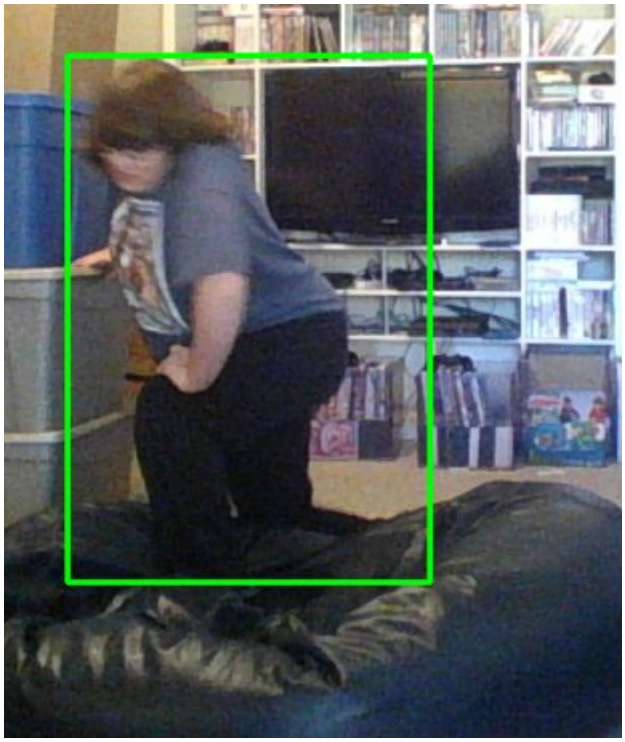
Interface design (with screenshots)

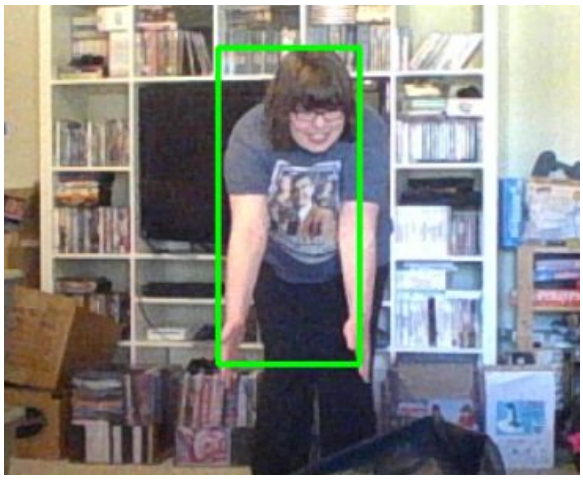
In order to allow for maximum flexibility across systems for the end-user, Vigilant is entirely accessed through a terminal environment. This gives the end-user the ability to quickly and easily add this system to a Linux server or any Unix-based system.

Implementation

As mentioned before the program tries to identify humans and draws a bounding box around them. The following examples were captured using a laptop camera.







Some of these bounding boxes are more accurate than others, but this is something that can be improved with time.

As you may have noticed the bounding box is green if the person is detected as standing and red if they are detected as having fallen over. Again, these images were captured using a laptop camera, so camera quality is unlikely to be the limiting factor. However, using a live feed, it does not run at a high framerate, and is constrained by the hardware doing the analysis. That said, by feeding different portions of the video to different computers, a much higher frame per second analysis could be done by running them in parallel.

Lessons learned

Through this project we have acquired experience with computer vision and specifically OpenCV. Our early research covered the various techniques for object detection and classification so that our code could identify people. This primarily involved edge detection and a built-in heuristic to identify shapes as human or otherwise. This same framework is useful in a wide variety of computer vision applications and is the basis of all object detection algorithms. The experience with OpenCV specifically is also widely applicable given that it is one of the most popular computer vision frameworks in use and it uses C++ which is easily integrated into most tech stacks.

Potential impact

The main focus of Vigilant is to aid existing video surveillance systems and provide more value by allowing it to assist in providing help to people in need. The development of the product has the potential to save lives, since the time it takes for people to receive aid once they have taken a serious fall is critical to how well they recover and/or survive. Many public and private areas are already heavily covered by surveillance cameras. These same systems can be augmented with our code to decrease response time in the event of a life-threatening incident whether the problem is the fall itself or a critical health incident that causes the fall such as a heart attack or seizure.

Apart from the potential to save lives, this software also has the potential to resolve lawsuits by identifying potentially critical footage. Often security footage is deleted by routine cleanup programs and lost before it is able to be subpoenaed by the interested parties. If any footage that is marked by our program is preserved for a longer than usual time frame this could help accident victims in pursuit of redress for damages.

One final use potential impact of this program is in assisted living situations. Commonly senior citizens are hesitant to go into dedicated retirement homes due to the loss of autonomy and the potential for abuse. The same concerns render some elderly people unwilling to live with a dedicated social worker. Our program, along with a comprehensive suite of cameras,

could help resolve this problem by allowing fall-risk people to continue living alone as they wish without overtly invasive assistance.

Future work

There are several avenues for improvement to our program. One example we considered during development was the automated creation of a video file containing five second clips of each detected incident of a fall. This would be useful for practical reasons in that it allows a human operator to quickly review fall cases and determine the proper response to filter out false positives and it would also be useful for legal purposes in that it provides a direct record of the most immediately concerning period and could be used as evidence in a court of law.

Another idea we had for improving the algorithm is classifying people in the lens based on their relative risk in order to better inform the decision about whether a fall is serious. If a person is relatively young then it is more likely that they will be only mildly harmed by a fall that would seriously injure an older person. It is a question of scraped skin against broken bones, and it is thus quite likely that younger people suffering from a serious fall might not need assistance.

Lastly one potential future avenue is developing a more traditional interface including GUI and the usual accoutrement. We believe there are definite advantages to maintaining a command line interface for the purposes of maximizing the compatibility of this project and enabling other developers to easily and quickly integrate this system into other projects. However, should our project be used as a standalone product it would be useful to have a more user-friendly interface for the purpose of increasing mass market appeal.

4.3 Risks

Risk	Risk Reduction
False - Positive False - negative	Optimize/ train fall detection algorithms with edge case scenarios
Adding Server Overhead	Optimize processes to use as little processing power as possible or determining a minimum hardware requirement

4.4 Tasks –

1. Research on computer vision models.

2. Selection of best model given research. Determining specific needs for the remainder of the project. (What type of data will we need? Will this be a supervised or unsupervised model? Will we keep a database? How much of an interface do we plan to build for users?)
3. Data collection and labeling.
4. Data manipulation and preparation.
5. Implementation of computer vision model.
6. Iterate upon model.
7. Interface for receiving model results.

4.5 Schedule –

Tasks	**Dates**
1. Research on computer vision models.	08/26-09/06
2. Selection of best model given research.	09/07-09/13
3. Data collection and labeling.	09/14-09/27
4. Data manipulation and preparation.	09/28-10/11
5. Implementation of computer vision model.	10/12-11/1
6. Iterate upon model.	11/2-11/15
7. Interface for receiving results.	11/16-11/29
8. Presentation prep and final report.	11/30-12/12

4.6 Deliverables –

- Design Document: A description of the components of our project.
- Model and components: If this ends up being a supervised model, any data used to train the model will be included in the final project submission.
- Final Report

[The final deliverable to the instructor is a zip file containing all reports and code. Also, all results from the project are posted on your website (except, optionally, any proprietary code).]

5.0 Key Personnel

Austin Kreulach – Junior computer science and mathematics major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Artificial Intelligence, Algorithms, Formal Languages, and Numerical Linear Algebra. Austin has developed an artificial intelligence that composes music, an artificial intelligence that detects malicious activity on a server, and a celestial navigation system for unmanned aerial drones. Austin is responsible for the Python A.I. architecture(maybe?), and literature review(background research).

Zane Turner - Junior computer science major at the University of Arkansas. He has completed machine learning and data mining which are somewhat relevant to our project. He had an internship with Cerner where he did machine learning related work. Responsible for helping build the algorithm and collecting some of the data.

Haven Brown – Brown is a junior Computer Science major in the Computer Science and Computer Engineering Department and Pure Mathematics major in the Mathematics Department at the University of Arkansas. She has completed Artificial Intelligence and Algorithms. She also completed an internship at Tyson Foods, Inc. in data science and machine learning. She will be responsible for gathering research on and developing a computer vision model capable of detecting falls in public spaces.

Lane Trobee - Senior, Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Computer Networks and Programming Paradigms which will prove to be of use in the project. He is interning with JB Hunt with a primary focus on front end development and user experience design. He will be responsible for user interface and user experience design.

Ian Moncur - Senior computer science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He interned at JB Hunt and is also pursuing a degree in Applied Mathematics. He has completed Algorithms, Numerical Analysis, and Numerical Linear Algebra. He will be responsible for gathering research and sample data.

5.0 Facilities and Equipment

No facilities or equipment will be required for this project.

7.0 References

[1] Caroline Rougier, Jean Meunier, Alain St-Arnaud, Jacqueline Rousseau, “Fall Detection from Human Shape and Motion History using Video Surveillance,” *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, Niagara Falls, Ont., pp. 875-880, 2007.

[2] Rita Cucchiara, Andrea Prati, Roberto Vezzani, “A Multi-Camera Vision System for Fall Detection and Alarm Generation,” Expert Systems, Wiley, 2007.

[3] Yue, Z., S. K. Zhou, R. Chellappa, “Robust Two-Camera Tracking Using Homography,” Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004.