Fall Detection

Gavin Burris, Logan Kilfoy, Logan Koch, Kyle Lepine, Mason Sanborn

*Abstract*—The issue of taking care of the elderly and the physically disabled has always been on our minds as we grow up and look to offer our parents and grandparents the same level of care that they provided for us. One of the worst situations for an elderly person to be in is to have fallen with no way of contacting help. Therefore, different methods have been created to prevent this situation from happening. While no solution is completely foolproof, we believe our project takes the next step towards ensuring the safety of those most at risk.

In this paper, we will study multiple examples of existing fall detection solutions as well as our findings for different approaches to fall detection. We will examine each of these approaches and ultimately determine which approach we deemed to be the best way to detect when people fall down and then get them the help they need. Through this study, we will get to understand the severity of this problem and find how our discoveries can be used to aid in the advancement of fall detection systems in the real world.

Keywords—Fall Detection, Databases, Templates, Haar Cascades, Contour Detection, Machine Learning, Motion Detection, Background Removal.

# Introduction

Our objective is to find and develop a better way to go about fall detection. The point of this project is to have a way to detect falls while being as unintrusive on the owner as possible. Products today like LifeAlert work as a necklace that does not detect falls but allows the owner to press a button and call for help when they fall down. In theory, this solution works, but it has two drawbacks. One, the product requires that you wear the device around your neck constantly, which can become an annoyance in day to day life. The other drawback is the means of detection. Because there is no means, if the owner falls unconscious when the fall occurs, there is nothing the owner can do to call for help. Our product works to combat these drawbacks.

The Fall Detection System will consistently and accurately detect when a person falls by means of vision processing techniques without being intrusive to the user. The system works by placing cameras in strategic positions around a home and then monitoring the environment for movement which is then processed to determine if a fall has occurred. Our system works in two parts through the detection of humans and then detecting when they have fallen.

Firstly, to detect when there is a human in the video feed, we use foreground detection methods to create a focus on the changes in the video feed and then contours to create a focus on the subject. Then a bounding box is created around the subject and a splice of the video is created and passed to the fall detection methods.

Pre-made templates will be the driving idea behind our method of fall detection. By comparing the subject’s current position in the spliced image to an array of templates from a database, our program can determine the similarity between the focused subject and its proper classification. If the classification meets the similarity threshold of a fallen human, a fall classification is created. The system then stores a history of the last several classifications and when the number of fall counts meets our threshold; a flag is thrown that signals a fall has occurred. This history of classifications is done to ensure that the majority of recent classifications are deemed as falls to prevent false positives from a possible misclassified frame.

Other fall detection products that are already in use can be divided into two categories, sensor-based detection, and vision-based detection. With our templating approach, we hope to follow more along the side of vision-based detection products by creating a program that works efficiently and accurately to detect when a fall has occurred.

# Machine learning approach downfalls

Another proposed way to detect falls using video-based fall detection is to use 3D cameras, such as Kinect sensors. 3D cameras provide depth, which many researchers prefer to use for fall detection. With 2D cameras, the system can cover more ground and view upwards of 15-20 feet. [2] Our system implements 2D vision and can detect falls within 25 feet of the system’s camera.

For fall detection, another approach is by utilizing machine learning techniques. [2] These techniques yield great results but can be costly on the machine hardware. Our system is aimed towards commercial use. With this in mind, we wanted to lower the cost of our system to become more affordable for average consumers. With lower-end hardware, a system with a machine learning implementation may not be feasible. But with our system implementation, creation of lower cost fall detection system may be possible.

# Socioeconomical Cost Associated With Falls

South Korea has an ever growing elderly population and the need for a fall detection system could save them lives and money. Estimates have indicated annual costs associated with elderly falls is around 1.3 trillion KRW(South Korean Won). The majority of the individuals who fell were female which indicates that a majority of the templates going forward should be female.

Falls were recorded in around 33% of the surveyed, this is in line with previously stated 30% annual fall rate. This is a global issues and our system is a step in the right direction to solve this problem.

# Template-Matching approach for User-aided Fall detection

The aging population of the world poses major concerns to governments and healthcare facilities since it will increase the demand for healthcare services and increase the burden on our healthcare systems. One strategy to prevent the elderly from requiring healthcare services is by implementing prevention and detection technologies in the homes of the group most at risk. [3] The necessary technologies vary from household to household, but the primary cause of injury-related death for the elderly is from falls.

Falling down is such a scary situation for the elderly to be in due to the lack of options. If an elderly person is living alone and falls, often they will not be able to get back up to call for help. As falling is the primary cause of injury-related death for elders, much research has gone human fall detection and prevention. Several works that appear in the literature offer different approaches to detect human falls. At their core, the two main detection styles are “environmental” (e.g., infrared sensors, pressure, microphones, and cameras) if they are placed in the environment or “wearable” (e.g., accelerometers) [3].

Both methods of detection have their own drawbacks but regardless of the detection method chosen, the need still exists. The approach presented in this journal by Diego Droghini, Daniele Ferretti, Emanuele Principi, Stefano Squartini, and Francesco Piazza uses a combination of the template matching approach we incorporated into our project and a sound-based element that listens to the room waiting to hear a loud thud that sounds different from the normal sounds that you could expect to hear in day-to-day activities. I personally like this approach as it adds an extra layer of security to the system. By using template-matching algorithms in combination with sound detection, you can root out many of the false positives that using one or the other would create. As described in the journal, their team of developers used templates of many false positions to balance out the templates of falls just like we did; however, they went on to create sound bites from general day-to-day activities that would not be considered falls and then included sound bites of actual falls which helps the system differentiate between the normal sounds and an actual fall. So, while the methods of detection were completely different, the approach to making them was the exact same and when used in conjunction with one another, there are much less false positives thrown.

# iot fall detection

As technology has evolved it has increasing made its way into our homes. Through IoT (internet of things) we have now have the capabilities of transforming a home into personalized smart sanctuary. These smart homes will be able to use smart devices to directly to help people prone to fall accidents and injuries. Every year over 30% people over the age of 65 experience a fall. A fall detection system in smart home devices could drastically improve the lives and safety of our seniors. Our system using fall templates could be used with smart home devices such as security cameras commonly sold at big retailers. These cameras could be connected to an overall security system or to a Google Home or Amazon Alexa. Template matching could help identify falls using these smart cameras and call for help in the event of a fall to help prevent more life threating injuries from occurring.



*Figure 1: Proposed Solution to IoT Fall Detection*

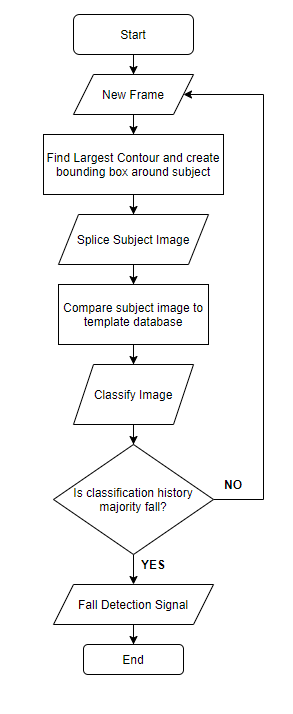
A proposed solution to an IoT fall detection system is shown in the figure above. This would utilize many different modules: Camera, Speaker, Server, and an Amazon Echo. Our system could be directly placed into the Fall Detector module and keep the rest of the system intact.

VI. Challenges, issues and trends in fall detection systems

Throughout the last several years many varying types of fall detection systems have been developed to counter the risk that a fall comes with. These systems vary drastically in approach varying from wearable sensors to fully manufactured environments with different types of sensors to provide the data necessary to properly calculate when a fall has occurred.

The major dangers presented when a fall occurs are not limited to the immediate dangers as many risks can occur from slow response time. More than 20% of patients admitted to hospital because of a fall had been on the ground for an hour or more, and even if there was no direct injury from the fall, their morbidity rates within 6 months were very high [4]. Because of this major risk our system also works to respond within seconds by quickly processing the live video feed of the home and raising a flag when a fall has occurred. This flag be handled to provide immediate support to the user with no action required of them ensuring an unassisted response that is not guaranteed by systems that require the user to establish when a fall has occurred.

# VII. Our solution



*Figure 2: Flowchart*

*A. Dynamic Kernel Size*

For our system to make accurate templates from processed video frames for subjects at a dynamic distance, we needed to come up with a way to adjust the morphological kernel size used to perform morphological closing operations. We approached this problem with the mindset of varying size subjects. Using the frame’s largest contour, we can find the width of the subject. Using the width, we can dynamically change the size of the kernel.

*B. Using Image Processing to Create Templates*

A detrimental component in the creation of a template-based fall detection system is the templates themselves. The goal in creating a template to be used for this system is to create a binary image out of the current frame of a live video feed where the only visible subject is the human in frame. We hypothesized that there would be two most valuable variations of templates we could create that would be helpful in classifying future human states seen in live video frames, templates derived from the extracted edges of a human subject and templates derived from the detected foreground, or subtracted background, of a detected human subject. The creation process of these two template types are quite similar.

For the templates derived from the extracted edges of the human subject, the first step is to extract a sub-image from the original image, the sub-image should exist inside the original image; however, it should only contain the human subject. Next, we convert the sub-image, derived from the most recent frame of the live video feed, to grayscale. Afterwards, we apply a Bilateral Filtering to the grayscale image, a Bilateral Filtering in image processing is an edge-preserving and noise-reducing smoothing filter. The Bilateral filter is similar to a Gaussian Filter and will blur the image to reduce noise present in the image; however, unlike the Gaussian Filter it preserves the edges in the original image. The Bilateral Filter can be defined by:

BF[I]p=1/Wp∑q∈SGσs(||p−q||)Gσr(|Ip−Iq|)Iq

After preprocessing the image, we then apply the Canny edge detection image processing technique to the bilaterally filtered grayscale image, to detect the edges of the current frame. This result gives crisp, clear results of edges visible in the frame of the live video feed, but the results contain more edges than just the human subject, these extra edges detected will be referred to as noise. To remove the excess noise of the image, we crop our extracted edges to contain just the edges that coexist in a foreground detected image of the original. The foreground detected image will serve as a mask in this case, and we will only be concerned with the edges that result after applying this mask.

After cropping the visible edges with the foreground detected frame, we resize the template to maintain fidelity in the comparison component of the system. In resizing our system uses a Bicubic interpolation, a non-adaptive image resizing algorithm. It considers a weighted average of a 4x4 neighborhood of pixels in the image to determine a pixel’s value in the resized image and iterates through every pixel of the new image to determine new pixel values. Since each image will originally be a different size it is impossible to determine the exact image reduction rate; however, each sub-image is resized to a 50x75 sized image.

Similarly, to the edge extracted template variant, the foreground detection template variant is created originally from a sub-image of the original live video frame, where its contents contain a fully encapsulated body of the human subject.

After the sub-image has been subtracted from the original image, the sub-image’s background is subtracted leaving just the foreground of the image using the background subtractor mog2 algorithm. Afterwards, the background subtracted sub-image is processed with a morphological closing algorithm using a dynamic kernel size specified in section *A. Dynamic Kernel Size*. The purpose of the closing algorithm is to fill in the subject’s silhouette, making them an easily recognizable figure to the Human State Classifier component of the system.

When the image has been processed it is then resized using the same Bicubic interpolation method used when resizing the extracted edges template type.

*C. Contour Detection of Movement*

The hardest hurdle that we had to overcome in this project was the question of how to detect people on screen. We tried many different approaches, including the use of Haar Cascades and HOG detection methods to classify objects in the video feed but both of these had their drawbacks. Using Haar Cascades was not a viable option because it was too taxing to process quick enough on a live video feed. Both Haar Cascades and HOG required machine learning training on consistent subjects in a stable environment. With the video feed being focused on home environments that are likely to be setup in a multitude of unpredictable ways we would be unable to train in a consistent environment. We also discovered that methods already using these approaches to detect humans did not do so when the posture and positioning of the human is subject to change drastically. With these setbacks in mind we set out on a different approach and began using contours as a means of focusing our detection.

First let’s define what a contour is. A contour is an outline of a curving or irregular figure. Our method of human detection revolves around finding the contours of the human and then drawing a bounding box around them. At the start of the program, we convert our video feed to grey scale and run a foreground detection background subtraction method. This method works by analyzing the full environment include any object or wall that the camera could see and comparing it to the previous frame with a set learning rate. The key is that we are only going to detect *moving* objects so by using this method, we are left with a greyscale video feed consisting of only objects that are non-static appearing in white on a black background. This allows us to draw contours over the white color to focus on any active objects in the video feed. After all the background noise has been eliminated, we look for the greatest contour being drawn using the *max* function seen in *Figure 3*. This function returns to us the contour with the greatest area which is likely to be the human in frame as they should be the largest active object in the environment.

Graphical user interface, text, application, email

Description automatically generated

*Figure 3: Codes for max contour detection and non-maximum suppression.*

So, in the end, our method for human detection did not end up being human detection at all. Due to the nature of this software and where it is going to be used, the most consistent method for our detection is to detect moving objects and process the focused frame to recognize when an object resembling a human appears to have fallen. The bounding boxes that we generate from our contour detection methods are essential to our product as the bounding box creates a splice of the video feed that is compared to hundreds of templates to ensure detection accuracy.

*D. Database*

There is only one database used in our Fall Detection System. The database was written in PostgreSQL. The database stores our templates and each template has five attributes: ID, type, characteristics, name, and the image itself. The images are saved as binary data and are all cropped to be the same size for uniformity. When the system is first started it will attempt to connect to the database, if it fails to do so it will try to load the templates from local files. This was done as the system was constantly tested on different machines and creating a database every time was not logistically ideal. The project will include local templates for use, so building a database is not required for the system to work.

*E. Non-Maximum Suppression*

Because of how crucial the bounding box selection is in creating the spliced image for template comparison we implemented non-maximum suppression methods from the numpy library. These methods allow us to take the rough values created from using the bounding box creation methods around the largest contour and create an average for them as seen in *Figure 2*. This average value bounding box is made with minimal processing time and ensures us a higher accuracy bounding around our subject.

*F. Classification of Human States using Templates*

The process of classifying the human states discovered in the frames of live video feeds is a detrimental component to the overall functionality and success of the system. The system relies on the templates created from the current frame of the live video feed; this process was specified in further detail in section *B. Using Image Processing to Create Templates.*

The system compares these templates with the templates stored in the local file structure, or the Fall Detection System database; these are both discussed further in section *D. Database.* The templates from the current live video feed, the extracted edges and extracted foreground, are classified using the K Nearest Neighbors algorithm. Since these two types of templates were created using different image processing techniques, the database and local file structure have been segregated to allow the system to use the two types as two different datasets to ‘train’ the K Nearest Neighbors model.

The K Nearest Neighbors algorithm implemented in our system takes a value k and training dataset to classify a given image as one of the classes that exist in the training dataset, and that the image is ‘nearest’ to out of ‘k neighbors’.

The algorithm runs such that the neighbors, N, for the image being classified, t, is initialized to the empty set N = Ø, and for each entry, d, in the dataset, D, that if the number of neighbors, |N| < k then N = N ∪ d; else if ∃ u in N such that distance(t, d) < distance(t, u), then replace d by the neighbor in N with the largest distance to t. Lastly, return c = class to which most u in N are classified.

Notable aspects of our implementation of the K Nearest Neighbors algorithm in our system is that we use weighted voting as opposed to majority voting. This means that rather than taking the mode of the Neighbors set, we take the max of the sum of the set of the reciprocal of the distances to decide a classification. Additionally, our implementation uses a max heap as opposed to a set for the Neighbors data structure.

# VIII. Performance Evaluation

|  |  |  |
| --- | --- | --- |
| **Testing Conditions** | **Bounding Box Inaccuracies** | **Human Detected?** |
| In the open, lights on. | 0 | Yes |
| In the open, lights off | 0 | Yes\* |
| Obstructed waist down. | 0 | Yes |
| Obstructed vertically. | 0 | Yes |
| In the open, 5 ft away. | 1 | Yes |
| In the open, 10 ft away. | 0 | Yes |
| In the open, 15 ft away. | 0 | Yes |
| In the open, 20 ft away. | 1 | Yes |
| In the open, 25 ft away. | 3 | Yes |

*Table 1: Human Detection Trials*

In Table-1, we show that our Fall Detection system has the capabilities of detecting a human in multiple different conditions. For these trials, we chose to test in optimal conditions, where the subject is in a well-lit room standing out in the open, in a poorly lit room, in obstructed situations both vertically and horizontally, and in multiple different distances away from the camera. The results above came from tests in a typical living room where each condition was tested for 5 minutes each. Our results showed that in every scenario, our system has the functionality to detect and draw a bounding box around the subject. With minor jumping of the bounding box in situations where the subject stops moving, the actual detection of the person is quite good but can have issues if you move farther than 25 feet from the camera.

These results were about what we expected. The Fall Detection system is designed to work around the most optimal and less optimal situations. Our implementation of the system works best when the subject is in open areas with the lights on and no obstruction, however strategies like our dynamic kernel size and our bounding box memory allows the system to adjust to scenarios where the conditions are not optimal. One important asterisk to note is that for human detection when the lights are off, the system was running using a camera with night-vision which allowed the system to detect the subject. When all lights are off without this night-vision, the camera will be blind and human detection is no longer possible.

|  |  |  |
| --- | --- | --- |
| **Testing Conditions** | **Average False Negatives During Fall** | **Fall Detected?** |
| In the open, lights on. | 3 | Yes |
| In the open, lights off. | 5 | Yes\* |
| Obstructed waist down. | 5 | Yes |
| Obstructed vertically. | 2 | Yes |
| In the open, 5 ft away. | 1 | Yes |
| In the open, 10 ft away. | 4 | Yes |
| In the open, 15 ft away. | 5 | Yes |
| In the open, 20 ft away. | 6 | Yes |
| In the open, 25 ft away. | 6 | Yes |

*Table 2: Fall Detection Trials*

In Table-2, we show that the Fall Detection system has the capabilities of detecting a human fall in multiple different conditions. For these trials, we chose to test in the same conditions as our human detection trials for consistency. The results above came from tests in a typical living room where each condition was tested 10 times with the average false negatives being rounded down. Our results showed good performance across the board in accurately detecting when the subject fell down. However, every test had false negatives thrown that continue to hinder our Fall Detection system from working perfectly. Our system uses a Human State Classifier to determine the person’s position every frame by comparing the subject’s bounding box and its contents to an array of hundreds of different templates, and as a result of this constant comparison, false negatives are prevalent in fast moving situations like a fall that result in imperfect data. Therefore, we determine if a fall is actually a fall if the majority of the last 20 frames were a fall. In all of these trials we consistently were able to report a true fall using this ruleset.

# IX. Conclusion & Future Work

In this paper, we proposed a different kind of solution to the fall detection problem. Using a template-based fall detection system, we were able to make a system that can classify people by position, and subsequentially tell if they have fallen down or not. We are able to detect these falls in a variety of different situations. This system will hopefully be able to prevent serious injuries from people who use it in their homes. Disabled people or the elderly in particular will benefit from having that extra level of security in their homes.

Some areas of improvement include reducing the number of false positives and negatives by adding an equal number of templates to each classification. The current system has hundreds of templates spread across each classification, but the distribution is not equal, and the number of templates could be higher which would result in a better system overall. Another area of improvement would be in our method of detection. As of now we detect people by finding the largest moving object in the room, but the system will not classify the person if they are not moving, so that could be modified to allow for constant classification.

In the future, we hope to have our system improved to the point where it is no longer throwing false negatives and positives. Once the system has been perfected, we hope that it will be used to help protect those most at risk of injury by fall and get them the help they need if they were to fall.

##### References

1. .
2. Huang, Z., Liu, Y., Fang, Y. and K. P. Horn, B., 2018. “Video-Based Fall Detection For Seniors With Human Pose Estimation.” IEEE 4th International Conference on UniversalVillage, Article
3. Diego Droghini, Daniele Ferretti, Emanuele Principi, Stefano Squartini, Francesco Piazza, “A Combined One-Class SVM and Template-Matching Approach for User-Aided Human Fall Detection by Means of Floor Acoustic Features”, Computational Intelligence and Neuroscience, vol. 2017, Article ID 1512670, 13 pages, 2017. <https://doi.org/10.1155/2017/1512670>
4. SM. Friedman, B. Munoz, et al. “Challenges, Issues and Trends in Fall Detection Systems.” BioMedical Engineering OnLine, BioMed Central, 1 Jan. 1970, biomedical-engineering-online.biomedcentral.com/articles/10.1186/1475-925X-12-66.
5. S. Greene, H. Thapliyal and D. Carpenter, "IoT-Based Fall Detection for Smart Home Environments," in *2016 IEEE International Symposium on Nanoelectronic and Information Systems*
6. Lee, Yeong Guk et al. “Complications and Socioeconomic Costs Associated With Falls in the Elderly Population.” *Annals of rehabilitation medicine* vol. 42,1 (2018): 120-129. doi:10.5535/arm.2018.42.1.120