Smart Vision for Managed Home Care

Jackson Dean Goodwin

Dept. of Computer Science

University of Tennessee at Chattanooga

Chattanooga, TN 37403

mwq755@mocs.utc.edu

*Abstract*—With an increasing number of elderly living alone, it becomes a challenge to ensure their safety and quality of life while maintaining their independence. The aim of this research is to investigate a new software system for pervasive home monitoring using smart vision techniques. In-home activity monitoring can provide useful information to doctors in areas such as behavior profiling, and preventive care. It can also facilitate in emergency detection and remote assistance. The proposed solution uses an inexpensive webcam and a computer program to analyze posture using techniques such as foreground detection and ellipse fitting. This paper will describe the merits and challenges of using computer vision as a solution to monitoring and evaluating daily activities of the elderly.

*Index Terms*—elderly care, activity monitoring, computer vision.

# Introduction

As life expectancy continues to increase, so does the demand for elderly care. Most elders would prefer to live in their own homes, but the gradual loss of functioning ability often causes these individuals to require some sort of assisted living or long term care such as that found in a nursing home. Pervasive home monitoring could afford elders a certain level of security and improvement in quality of life while maintaining their independence and privacy. Low-cost vision-based systems can be used to monitor and evaluate daily activities of the occupants.

The ultimate goal of this research is to develop methods for analyzing the motion of an individual in a scene in order to extract information about the person’s posture, behavior, and activity. Major research questions include how to separate the individual from the scene, how to detect posture, and how to ensure privacy. In this paper, we focus on methods for detecting foreground, detecting humans in the foreground, and detecting posture, so that we can monitor activity and behavior.

# Motivations

Individuals with poor postural stability who do not need assistance with activities of daily living can benefit the most from a low-cost vision-based home monitoring system. Such a system would allow these individuals to remain at home and continue regular activities without incurring expenses and loss of privacy or independence from hiring a healthcare professional to be in the home.

The cost of providing such a monitoring system is less than the cost of hiring a caregiver or living in a nursing home. Costs for care range from $70 a day at an adult day health care center, to $200 a day for a semi-private room in a nursing home. Costs for an in-home pervasive monitoring system would be a one-time investment of about $500 for several wireless cameras (and possibly other devices beyond the scope of this project), and $1200 for a computer base/processing unit. Then there would be an additional cost of about $30 per day. The up-front cost for a monitoring system would be more, but it would match the price of adult day health care in 45 days (in 25 days for assisted living and in 12 days for a nursing home).

Monitoring systems can also provide more privacy for the occupant for two reasons: 1) The occupant can continue to live alone without disturbance, and 2) Identifying features from the cameras can be removed programmatically. The computer does virtually all of the analysis, so the only time anyone would access images from the cameras would be in an emergency that required further examination of the situation.

# Related Work

Researchers at the Wireless Sensor Networks Lab at Stanford University have developed a system [[1](#_ENREF_1)] for monitoring elderly persons remotely. They make use of a wireless badge containing accelerometers to detect a signal indicating a fall. Wireless signal strength is used to triangulate an approximate location to trigger cameras with the best views when a fall occurs. Each camera processes the scene independently to identify whether a human body exists in the image. The scene is first analyzed to detect motion events through the use of background subtraction and blob segmentation. Blobs are further classified as human or non-human by analyzing the percentage of straight edges and skin color in each blob. If the blob is a human, the posture and head position is estimated and a certainty level is reached by each camera based on the consistency of the obtained results. Posture is analyzed by fitting an ellipse on the blob and analyzing the orientations of the major and minor axes. The head is detected using skin color or shoulder-neck profile or both. Through collaborative reasoning among several cameras, a final decision is made about the state of the user. This work addresses the issue of reliability of results by making use of different camera views, reducing the number of false alarms and increasing efficiency.

Tarik Taleb et. al. present a framework [[2](#_ENREF_2)] for assisting elders at home called ANGELAH, which is a middleware solution integrating elder monitoring, emergency detection, and networking. It enables efficient integration between a variety of sensors and actuators deployed at home for emergency detection and provides a solid framework for creating and managing rescue teams composed of individuals willing to promptly assist elders in case of emergency situations. This work discussed issues relating to elder support group formation.

Researchers at the Imperial College London have developed a ubiquitous sensing system [[3](#_ENREF_3)] for behavior profiling. This system uses vision based activity monitoring for activity recognition and fall detection. To circumvent privacy issues, their vision-based system filters captured images at the device level into blobs, which only encapsulate the shape outline and motion vectors of the subject. To analyze the activity of the subject, the position, gait, posture and movement of the blobs in the image sequences are tracked. This subject-specific information is called personal metrics. The basic goal of their system was to measure variables from individuals during their daily activities in order to capture deviations of gait, activity, and posture to facilitate timely intervention or provide automatic alert in emergency cases. The posture is estimated by fusing multiple cues, such as projection histogram and elliptical fitting, obtained from the blobs and comparing with reference patterns. From the posture estimation results, activity can be accurately determined. This system can also determine different types of walking gaits and decide whether the user has deviated from the usual walking pattern. The main issues covered in this work are ensuring privacy by filtering captured images at the device level, and using personal metrics to profile behavior.

void calibrate()

{

convertToHSLColorSpace

(the frame from the camera);

convertToAFloatingPointScale

(the image from the previous line);

findTheSumOfTheAccumulatedFramesAnd

(the current frame);

make a backup of the background model;

if (it is time to check the exposure)

{

detectForeground(); // see figure 3

detectSkin();

determinePercentageOfForegroundIn

(the foreground image);

if(the percentage of foreground in the

image > the threshold)

{

Revert to the old background model;

stop calibrating;

}

else  
 {

increase the exposure;

if(we have gone through all of the

exposures)

{

stop calibrating;

}

}

}

}

# Proposed Approach

The focus of this research is on behavior profiling through the use of smart vision techniques. First, posture must be detected and then changes in the posture over time can be analyzed to detect events and evaluate activity. We wrote a computer program that detects posture in real time from streaming video. Much of the image processing was done using the algorithms already provided in the OpenCV library. The input for this program is provided by a camera, and the output is intended to be a description of the person's activity. The activity is in terms of how much time was spent standing up, walking, sitting down, lying down, etc. This system could also be used to alert caregivers to a fall. So far, the program only differentiates between three activities (standing, sitting, lying down) and fall detection is not implemented.

***Foreground Detection.*** The first step was to separate the foreground from the background. Generally, the foreground is everything that is either moving or new, and the background is everything else that does not move and does not change. Foreground is the part of the image that is of interest, since it is the only place where a human should appear. Figure 1 illustrates the concept of foreground and background.



1. A classroom serves as the background (*left*). The author becomes the foreground when he enters and stands in the room (*center*). The foreground is shown in white, and background is shown in black (*right*). Note that the shadow is also detected as foreground.

There are several ways to detect foreground. One way is to simply subtract each frame of video from its immediate predecessor. This is called frame differencing. This method has its drawbacks as it only works for detecting objects that are currently in motion, and it generally only detects the outlines of objects. A more sophisticated way to detect foreground is to create a model of the background and compare images from each frame to the background model. This is the method that we have implemented.

In order to create the background model, our program first goes through calibration. This process lasts about 15 seconds, and creates several background models, each corresponding to a different exposure level of the camera. This allows for the program to adjust to changing lighting conditions and to automatically choose the best exposure for the scene at any moment.

1. Pseudocode for the calibration algorithm

Sometimes the background model becomes invalid. Such is the case when the lighting in a room changes and when the camera is moved. Other times, the background may be valid, but there is no detected foreground (i.e. no one is in the scene). In this case, it would be useful to recalibrate the scene in case there have been any changes to the background since the person left. For these reasons the program recalibrates when either: a) There is not much in the foreground (which implies that there is no one in the room), or b) there is too much in the foreground (which implies that the lighting in the room has changed, the camera has moved, or the background model is no longer accurate).

Once we have a background model, we can detect foreground by finding the difference between the model and the current frame of video. This is done by finding the absolute value of the difference between the components of the model and the components of the current frame, and then applying a threshold. The threshold is used to make a binary image (i.e. one whose pixels are either white or black – foreground or not foreground) out of the grayscale image created from the subtraction of the frames. This process uses the same technique as the frame differencing method mentioned earlier, but differs from this method in that the current frame is compared to the background model instead of the immediately preceding frame. Once the foreground is detected, an algorithm is applied to the image to remove noise (i.e. isolated specks and unwanted small regions) and clean up the edges. This works by applying opening and closing morphological transformations (which use dilation and erosion operations) and by removing contours that are too small to be important.

1. Pseudocode for the foreground detection algorithm

Also, worthy of noting is the fact that we use the HSL (Hue, Saturation, Lightness) color space for the foreground detection. The reason for this has to do with the difference between how computers and humans see color and intensity of light. As it applies to foreground detection, the RGB (Red, Green, Blue) color space is very sensitive to changes in brightness. Therefore, applying a threshold to each of the channels in the RGB color space tends to detect shadows in addition to detecting real foreground. Using the HSL color space reduces this effect to a certain degree, but does not completely eliminate it (see Figure 1 (right) and note).

Regardless of the color space used for foreground detection, one challenge is dealing with the fact that when a part of the foreground is similar in color to the corresponding part of the background model, it is less likely to be detected as foreground. This can be seen as black lines or shapes appearing where there should be white foreground.

***Human Detection.*** The next step is to determine whether or not the foreground “blobs” (i.e. connected regions in the foreground) are human contours. Human blob detection can be done by combining the results of detecting the head, the percentage of skin color, and the percentage of straight lines in the blob. Each of these factors can be weighed to determine whether the blob is a human.

The easiest technique to implement is skin color detection. This can easily be done using the HSL color space by applying two hue thresholds, an upper and a lower, to define a range of hues that corresponds to the color of skin. (Hue is a term that refers to what color something is e.g. red, orange, yellow, green, cyan, blue, and corresponds to the frequency, or inversely the wavelength, of light that it appears to be. This is in contrast to saturation, which means closeness to a pure color versus a shade of gray, and lightness, which is the perceived intensity, or brightness of the color.) If a blob has a large enough percentage of skin color (from the original image), then it has a greater chance of being a human.

void detectForeground()

{

convertToHSLColorSpace

(the frame from the camera);

splitIntoIndividualImagesForEachChannel

(the frame from the camera);

splitIntoIndividualImagesForEachChannel

(the background model);

findTheAbsoluteValueOfTheDifferenceBetween (each channel of the frame and model);

applyAThresholdTo

(the differences of the frames);

convertFromRGB2GRAY

(the frame from the camera);

combineSingleChannelImagesIntoOneImage

(each of the channels from the frame);

applyAThresholdTo

(the image from the previous line);

removeNoiseFromAndSmooth

(the image from the previous line);

}

1. Pseudocode for the skin detection algorithm

void detectSkin()

{

for(each pixel in the frame from the camera)

{

convertToHSLColorSpace

(the frame from the camera);

if

(the color is within the range of hues

and saturations for skin color)

mark the pixel as skin;

}

}

The next technique we implemented was the straight line detection. The idea is that manmade objects tend to have straight-line components, while humans do not. Application of a threshold to the number of lines detected in the part of the original image that corresponds to the blob can be used to determine whether the blob is likely to be a human. The Hough line transform is used to detect lines in the blobs.

1. Pseudocode for the line detection algorithm

void detectLines()

{

detect edges in the frame from the camera;

detect lines that lie within the foreground region with the Hough line transform;

if(there are fewer than x lines in the

foreground)

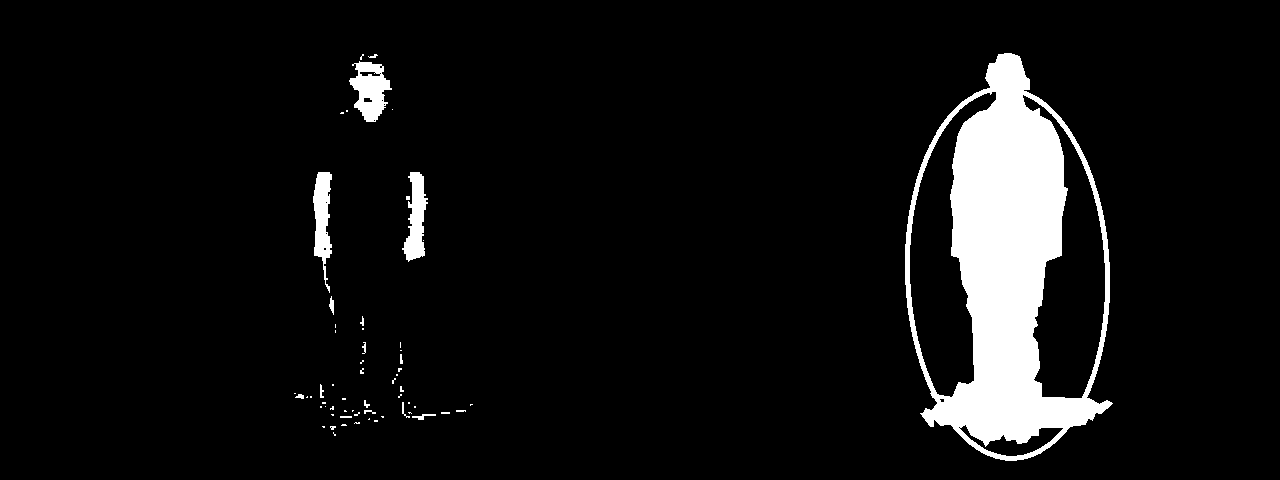
increase the probability that

the blob is a human;

}

The last technique for human detection is head detection. This is the most complex and probably the most decisive factor in detecting a human body. There are several ways to detect a head. One way involves skin color and the position of the head relative to the body, another involves the shape profile of the head, neck, and shoulders, and another involves complex analysis of facial features. The technique that we implemented involved fitting an ellipse around the blob to find the ends of the body (i.e. the ends of the major axis of the ellipse), and then calculating the percentage of skin color in a small region around each end. This relatively crude solution detects the head most of the time when the blob is a human, but can fail when the person is not facing the camera. Figure 7 depicts a couple of the steps involved in human detection.

1. Pseudocode for the head detection algorithm



1. Skin color detected in the blob (*left*). Ellipse fitted around blob to locate the head (*right*).

***Posture Detection.*** Next, posture can be determined from the lengths and orientation of the axes of the ellipse fitted around the blob. The algorithm that we implemented to do this is very simple. If the ellipse is taller than it is wide, the person is either standing or sitting. If the ratio of the longer axis to the smaller axis is below a certain threshold, then the person is sitting. Otherwise, the person is lying down.

# Results

The foreground detection will still detect shadows, especially if one stands near a wall or some other vertical surface. The detection of shadows will usually cause the program to report that the user is sitting down when he is really standing up near a wall. Also, extending the arms out to the side will change the shape of the ellipse being fitted around the body, and will cause the program to report that the person is sitting down. Occasionally, objects in the background that are about the same color as the object in the same position in the real foreground will not show up in the detected foreground, because the difference in color does not exceed the threshold. The human detection algorithm is rudimentary and does not always filter out non-human objects.

# Future Work

Much more can be done in the way of human detection, especially with the head detection. A shoulder-neck profile algorithm or a facial feature detection algorithm would yield much more accurate results, but would also be more difficult to implement. Optimization of the HSL color space or use of another color space may produce more accurate results as well. A more robust foreground detection algorithm could better handle slight changes in lighting, such as that caused by changes in ambient light as from the sun on a partly cloudy day. Foreground detection could be improved again by not allowing shadows to appear in the foreground. Finally, more work can be done in the areas of fall detection, activity recognition, behavior profiling, and gait analysis to make use of the posture information.

void detectHead()

{

detect the contour of each blob;

fit an ellipse around the contour;

find the ends of the major axis;

detect the percentage of skin color in a

small region around each end;

if(that percentage > x)

a head has been detected;

}

More can be done to have a quantitative analysis of the results as well. Most of the data can best be interpreted through visualization. A confusion matrix can be used to visualize the performance of this algorithm.

# Conclusion

The three steps that must be performed before activity can be analyzed are 1) foreground detection, 2) human detection, and 3) posture detection. Each of these steps can be taken in various ways. We spent most of our time working on foreground detection. As a result, this is the most developed part of our program. The other steps are no less important, and can be developed in future work.

##### Acknowledgment

We would like to thank our professor, Dr. Yu Cao, for providing the opportunity for us to do this research and for answering our questions and guiding us along the way.

##### References

[1] A. Keshavarz, A. M. Tabar, and H. Aghajan, "Distributed vision-based reasoning for smart home care," 2006.

[2] T. Taleb, D. Bottazzi, M. Guizani, and H. Nait-Charif, "ANGELAH: a framework for assisting elders at home," *Selected Areas in Communications, IEEE Journal on,* vol. 27, pp. 480-494, 2009.

[3] B. P. L. Lo, J. L. Wang, and G. Z. Yang, "From imaging networks to behavior profiling: Ubiquitous sensing for managed homecare of the elderly," 2005.