Distributed Smart Cameras for Aging in Place*

Adam Williams, Dan Xie, Shichao Ou,
Roderic Grupen, Allen Hanson, and Edward Riseman
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{apw, dxie, chao, grupen, hanson, riseman}@cs.umass.edu

Abstract

This paper describes the design and preliminary implementation of two distributed smart camera applications: a fall detector and an object finder. These functions are part of a novel suite of applications being developed to address "aging in place" health care technologies. Our approach to these applications is unique in that they are based heavily on video data, whereas other solutions may require devices that must be worn or attached to objects. The fall detector relies on features extracted from video by the camera nodes, which are sent to a central processing node where one of several machine learning techniques are applied to detect a fall. If a fall is detected, alerts are triggered both in the home and to a third party. The object finder similarly uses a boosted cascade of classifiers to visually recognize objects either by request of the user or automatically when an object is moved.

Keywords

Smart cameras, activity monitoring, object recognition, aging in place, distributed sensor networks

1 Introduction

The growing numbers of elderly individuals in need of support to live in the community will severely test the current services infrastructure. Part of the solution is to develop technology for "aging in place," so as to creatively increase the length of time elders can remain at home. In the long term, the goal is to "consumerize" health and wellness technologies and make it practical and affordable to incorporate them into existing homes and lifestyles. Distributed sensor networks are well suited for this purpose. Sensors can be placed throughout an elderly person's living space to create a smart environment that monitors their safety and provides a variety of other services. Camera sensors are particularly attractive and cost effective due to the wide range of applications that can be based on visual data, including human tracking and object recognition. We are especially interested in the use of smart cameras, since their use will allow a reduction in costs and support requirements over those associated with PC-based cameras.

*This work supported in part by National Science Foundation grant SES-0527648, ARO grant W911NF-05-1-0396, and NASA grant NNJ05HB61A-5710001842.

1.1 Project Overview

This paper describes a work-in-progress distributed sensor network that provides health and wellness services to the elderly. The primary goal is to create a practical, unobtrusive, cost effective system that specifically addresses the special needs of the elderly. To this end, feedback and guidance are being sought from our target audiences throughout the development cycle. This feedback process is being facilitated by the Smith College School for Social Work through a series of focus groups of the elderly, their caregivers, and their families. We are prepared to adapt our applications, or develop completely new ones, based on the results of these focus groups. The initial set of applications includes a fall detector, object finder, video phone, calendar, and address book. We will focus our discussion on the fall detector and object finder applications, as they are the two that rely on the sensor network and are suited for implementation using smart cameras.

1.2 Application Overview

Surveys of caregivers and family members of the elderly have shown that knowing when an elderly person has fallen is one of their primary concerns [13]. This is the case not only due to the obvious, immediate medical attention that a fall may require, but also because frequent falling and instability can be a sign of more serious ailments. Two types of commercial products exist that are worn by the person and can alert a third party if a fall occurs. The first is a device that is worn around the neck and has a button that can be pressed to put the person in verbal contact with a dispatcher, who can alert the appropriate authority [7]. The major drawback to this device is, of course, that the fallen person must be conscious enough to press the button after the fall. A second device is also worn around the neck, but relies on a built-in accelerometer to automatically detect a fall [15]. As with the first device, this one will also trigger an alert to a third party, who will relay it to the appropriate emergency authority. In both cases, the user must remember to wear the device at all times in order for it to be effective.

Our system uses cameras and machine learning techniques to avoid the major problems with these commercial devices. Cameras are used to visually track a person as they move about their home. At each frame of video, several features of the blobs being tracked are extracted and used to determine if a fall has occurred. If a fall is detected, the system attempts to initiate a video phone call with one of several

previously determined emergency contacts. The contact is able to view the severity of the situation and act accordingly. While this method does require that the fall occur in view of a camera, it also relieves the user of the encumbrance of wearing a device, while still providing automatic alerts and enhanced communication to a third party.

The aging are often faced with short- and long-term memory deficiencies that negatively impact their quality of life. A common and particularly frustrating side effect of this is an increased frequency of misplacing household items such as keys, eye glasses, remote controls, etc. We are developing an "object finder" application to help with this problem. The user is able to initiate a visual search of their living space for a particular object, and the system will attempt to locate the object using detection and recognition techniques. If the object is found, the application will display a live view of its location. Most other object locating strategies involve physically attaching an RFID tag or sound-emitting device to important objects, which can be inconvenient or obtrusive. These other methods do have some advantages however, such as being able to locate an object out of view of any camera. We may incorporate RFID-based object localization into the system in the future.

2 System Overview

In this section, we describe the assumptions and general sensor requirements of our system. We also describe the specifics of our current prototype system. The primary design goal is to minimize the cost and effort of installing and maintaining the system, while ensuring that the applications still perform acceptably.

2.1 Camera Nodes

We assume that any living space that the user wants to be actively monitored by the system are in full view of one or more camera nodes. Each camera node consists of a camera sensor, a CPU, RAM, and a network connection. An example of a node could be a PTZ camera connected via firewire to a standard PC with a wired ethernet connection, or a wireless smart camera with an embedded CPU and RAM. As previously mentioned, wireless smart cameras are preferable due to their low profile and relatively low cost. High-resolution sensors are not required for either of the applications discussed in this paper. Fall detection performs effectively with a resolution of 128x128, and object recognition with 320x240. The low resolution requirements result in low CPU and RAM requirements as well, making them suited for implementation using embedded processors. The cameras do not have to be calibrated, although doing so allows additional functionality. Each camera node is responsible for extracting necessary data from its video stream and sending it to the central processing node. This decreases the processing latency and the inter-node bandwidth requirements. Each node can also act as a raw video server when necessary.

2.2 Central Processing Node

The system must have one central processing node that gathers the information produced by the camera nodes and handles CPU-, RAM-, and storage-intensive processing tasks. The central node handles resource management, dispatches fall alerts, and initiates object finding. The exact processing requirements of the central processing node depends largely on the number of camera nodes that exist in the network.

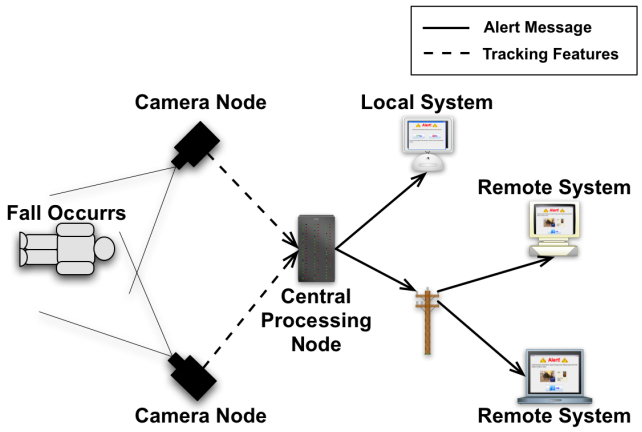

Figure 1. The fall detection and alert process

2.3 Other Devices

We also assume that the user will have a display, input device, and speakers in most or all major living spaces in order to interact with and receive information from the central processing node. We are particularly interested in alternative input methods such as touch screens and voice or gesture recognition. In the future, additional sensors may also be incorporated into the network, such as RFID for object localization, floor-mounted vibration sensors for fall detection in privacy-sensitive areas, and infrared cameras for additional tracking and health monitoring capabilities. We are also looking into the use of small mobile robots that can deliver services throughout the home.

2.4 Prototype System

Our experimental living space consists of five camera nodes. Each node is equipped with a Sony EVI-D100 PTZ camera mounted on the wall and connected to a VMIC single board computer via a Leutron Vision frame grabber. Each VMIC board has a 928MHz CPU, 256MB of RAM, and 128 MB of compact flash storage. The central processing node is also a VMIC board, but it has 30GB of hard disk storage. All of the nodes are connected via 100Mbps ethernet. Data is transmitted among the nodes via the NDDS publish-subscribe middleware. A simple TCP-based client/server model can also be used. This prototype system was originally developed with high-performance robotics applications in mind and has processing power that is in excess of the current applications' requirements.

3 Fall Detection

The fall detection application constantly tracks people as they move about the living environment in view of the cameras. At every frame of a person's motion, several features are extracted and fused with features from previous frames. This sequence of features is analyzed using one of several techniques to determine if a fall has occurred. If a fall is detected, the alert procedure is initiated.

3.1 Tracking

A simple tracking procedure is employed at each camera node and is suited for implementation on embedded processors. Specifically, a background subtraction method is used to extract foreground objects in each frame of video. The background model is a running average computed at each pixel [3]. When a new frame is captured, pixels in the frame that differ from the background model by more than a fixed threshold are considered to be in the foreground. The new frame is then averaged into the background model, with foreground pixels weighted less than background pixels. A dilation operation is then applied to the foreground mask in order to merge blobs that are very close to each other, followed by a fast connected components procedure to label the foreground objects. Objects that do not contain a large enough number of pixels to be a human are discarded. Then the 2D image coordinates, 2D velocity, aspect ratio, and color histogram of the human-sized blobs are extracted and sent to the central processing node for further analysis.

If a person is visible simultaneously in more than one calibrated camera, the central processing node may be able to match the blobs in different cameras using prior knowledge of camera location and the blobs' color histograms. If a match occurs, triangulation can be performed to provide 3D world coordinates of the person, which can make fall detection more effective. This information can also be provided to the emergency contact if a fall is detected.

3.2 Detection

The central processing node collects all of the tracking information provided by each camera node at each frame of video. The central node will attempt to compute frame-to-frame object correspondences for the blobs from a given camera node based on the color histograms and locations of objects. Using the frame-to-frame correspondences, a sequence of features is produced for each person in each camera. If 3D tracking is enabled as mentioned in the previous section, the correspondences yield one global sequence of features for each person. Regardless of whether the position and velocity features are 2D or 3D, several procedures are available for analyzing a sequence to detect a fall.

The first procedure is to simply make a decision based on the aspect ratio of the bounding box around a blob in the current frame. If the aspect ratio is greater than 1, the system assumes the person is horizontal and has fallen. Note that aspect ratio is defined as width divided by height. If 3D position information is available, the system can also determine if the person is on or near the floor. This procedure is prone to false alarms, particularly if someone is purposely sitting or laying on or near the floor, or if the foreground segmentation is inaccurate. This procedure is only recommended as a solution when CPU resources are limited.

A different and very effective supervised learning approach involves the use of Hidden Markov Models (HMMs) [12], a popular tool for activity recognition and fall detection [10] [4][14]. However it also requires large amounts of view-dependent training data that makes it impractical for large-scale use. Training data is collected by having a person perform several different common actions such as walking and sitting along with simulated falls in view of each camera. Tracking is performed as described in section 3.1, and each frame is labeled by hand as containing a fall or not. Each sequence of frames corresponding to one action is also labeled as containing a fall or not. A support vector machine (SVM) classifier [16] [2] is constructed using the individual 2-class labeled frame data. Two HMMs are also trained using the labeled sequences, one with falling sequences and one with no-fall sequences. Note that during the HMM training stage, each time step in a sequence has only one feature: a boolean value indicating whether the frame contains a fall. To detect a fall in a new sequence of activity, each frame is classified by the SVM as containing a fall or not. Once enough frames have been observed, the sequence of SVM decisions (one for each frame) is then evaluated under each HMM to produce two likelihood values. The sequence is then given the label of the HMM that yielded the maximum likelihood. In our experiments, this method detected falls with 98% accuracy, but again the effort needed to collect training data makes it impractical.

Finally there is an unsupervised technique that is the most practical of the three. This procedure only requires a collection of "normal" sequences of activities from each camera. These can be automatically gathered in an initial training period for the system, during which the system assumes a fall does not occur. Once these sequences have been collected, they are used as training data for a single HMM. As a new sequence is being observed, it is evaluated under this HMM at each new frame, which allows the rate of change of the normalized likelihood of the sequence to be monitored. If the normalized likelihood drops rapidly, something unusual is likely occurring, and some simple tests can be applied such as in the first procedure to determine if it is indeed a fall. This method shows the most promise in terms of practicality and qualitative effectiveness.

3.3 Alerts

The central processing node will dispatch alerts once a fall is observed. First, a local alert is broadcast over the speakers and display devices in the person's home. This provides reassurance that action is being taken in the case of a fall, or gives them a chance to cancel the alert in case of a false alarm. If the alarm is not disabled within a brief period of time, the central processing node will make attempts to display an alert remotely to the people on a prioritized list of emergency contacts.

It will try each contact until it reaches the end of the list, at which point it will contact 911 or other emergency service. The remote alert contains information about who has fallen as well as the location of the fall and a live video window of the area where the fall occurred. At this point, the contact can initiate a video phone call to the fallen person in order to make voice and visual contact with them so an assessment of the situation can be made.

3.4 Privacy

We must be very sensitive to the privacy concerns that will undoubtedly arise with the thought of placing cameras in people's homes. This is especially true in this case, as we are proposing to display video of people in vulnerable situations to their relatives or trusted friends. To help alleviate concerns

Figure 2. A remote fall alert

Figure 3. The object finder result display.

about this issue, a privacy guard setting is included in the fall detection system that will place a featureless silhouette over the image of a person in the live video that is streamed to the emergency contact. Work is also in progress on providing a virtual reality-based view of the person's home based on the technology described in [11]. The feedback received from focus groups of elders will guide us as we continue to deal with privacy issues in this and other aspects of the project.

4 Object Finder

The object finding application consists of two cooperative modules: Object Change Detection and Active Object Searching. The former monitors the scene and records the position of an object if it is moved. The latter is activated when the position of a queried object has not been previously recorded by the Object Change Detection module. These two components work cooperately to make the object finding process efficient and robust. To the extent of our knowledge, our object finding application is the first that combines object detection and active searching to allow people to find objects in their living environment.

4.1 Object Modeling

Two kinds of features are used to represent the object and perform the detection and recognition.

(1) *SIFT features*. SIFT (Scale Invariant Feature Transform) represents an image as a collection of local feature vectors, each of which is invariant to image translation, scaling, rotation, and partially invariant to illumination changes and affine or 3D projection [8] [9]. Recognition performed with this type of feature has been shown to be quite robust in realistic and complex environments.

(2) *Haar-like features*. The algorithm proposed in [17] [6] and provided by the Intel OpenCV libraries can achieve rapid object detection based on a boosted cascade of simple Haar-like features. These features are scale invariant and can be calculated rapidly. Increasingly more complex classifiers produced by the AdaBoost algorithm are combined in a cascade structure, which allows many background regions to be discarded quickly and is very effective for our application.

4.2 Object Finding Procedure

(1) *Object Change Detection*: The Object Change Detection module monitors the scene and launches the object detection/recognition when the position of an object changes. To achieve this, all of the cameras nodes continuously maintain the average background model and perform foreground segmentation as described in section 3. If a foreground blob (object) has been static for a given time T, it is considered a newly moved object. Information about the blob is sent to the central processing node where an attempt will be made to recognize the object. Since the object blobs are usually too small (10-20 pixels) for recognition, the camera will bring the object to the center of image and zoom in to a larger scale. Normalized cross-correlation and neighborhood region searching are used to match the object blob in frames when a camera is panning or tilting. In the recognition phase, the SIFT keys are calculated for the object blob and an attempt is made to match the object with those in the database. A match score is generated by the central processing node for each camera view. The sum rule for decision fusion [1] is used to merge the information from multiple views and generate a total score. If the score of an object is higher than a threshold, the position information (either 2D or 3D depending on the number of calibrated cameras that saw the object) will be added to this object's known position list for future queries. Object Change Detection reduces the time of the entire object finding process by storing the position information for any object it has seen.

(2) *Active Object Searching*: This module runs on the central processing node and directly handles the query from the user. It first checks the queried object's known position list, and if the position of the queried object has been recorded, it is reported to the user; otherwise the system executes a scan over the scene. The central processing node selects a scanning camera node, which generates a series of images (4-5 for a 10x10 room) that cover the scene and are sent back to the central processing node. Then the SIFT and Haar-like features are both used in an attempt to detect the queried object in these images. In most cases, the object detection algorithms using SIFT and Haar-like features both generate some candidate regions. A weighted decision method is used to determine the scores for all candidate regions, and we report any regions whose score is higher than a threshold to the user. The results are presented in the form of labeled images

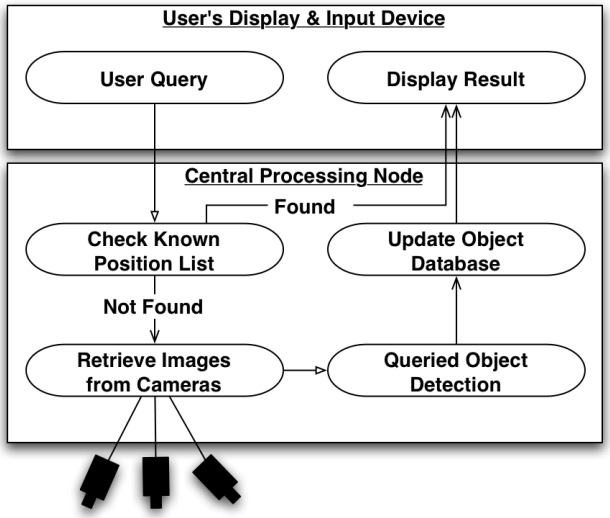

Figure 4. Active object searching

and text and voice descriptions.

To evaluate the performance of the object finder, we built models for 5 objects: a key, cup, cell phone, TV remote, and a book. The average success rate of the Object Change Detection module is 80%-90% and that of the Active Object Searching module is 70%-80%. The computation time is 1-2 seconds for Object Change Detection and less than 20 seconds for Active Object Searching.

5 Resource Management

While much of our work on resource management is left for the future, we discuss it briefly here because of its importance to distributed sensor networks. There are two primary, limited resources that must be managed: cameras and energy. Different applications may require the camera nodes to be in different configurations. For example, the fall detection may need tracking information from a camera node, which is also receiving a request to scan an area for the TV remote. Certainly detecting falls must take priority over finding the remote, so the central processing node will block the needed camera nodes from receiving lower priority requests. The camera nodes that will be needed for tracking by the fall detector can be determined based on the dynamics of the person's motion [5]. Similarly, this information can be exploited for duty-cycling to decrease the power consumption of the sensors.

6 Conclusions

Distributed smart camera networks for the aging in place of the elderly show great promise in their versatility, space efficiency, and cost-effectiveness. In this paper, we have described the prototype implementation of a fall detector and an object finder, two pieces of a larger suite of applications and services for the elderly that are in development. While the camera nodes in this prototype system contain PC-based cameras, we describe how some of the processing tasks required by the applications are well-suited for implementation on smart cameras. In the future, we plan to incorporate wireless Agilent Cyclops smart cameras into the system. This will require extensive work on energy management for efficient duty-cycling of these battery-powered devices, as well as additional work on managing heterogeneous camera nodes with varying resolutions and processing capabilities. In addition, the system must be reliable and perform predictably, since it will be providing critical services to its users.

7 Acknowledgments

This work is supported in part by NSF grant SES-0527648, ARO grant W911NF-05-1-0396, and NASA grant NNJ05HB61A-5710001842. We would like to thank our undergraduate researchers, Jessica Krause and Joe Gallo, for their programming work. We would especially like to thank Joan Hanson for her patience and willingness to fall on the floor repeatedly.

8 References

[1] R. Brooks, P. Ramanathan, and A. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8):1163–1171, 2003.

[2] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, November 1995.

[3] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[4] J. Gao, A. G. Hauptmann, A. Bharucha, and H. D. Wactlar. Dining activity analysis using a Hidden Markov Model. In *International Conference on Pattern Recognition*, 2004.

[5] D. Karuppiah, R. Grupen, A. Hanson, and E. Riseman. Smart resource reconfiguration by exploiting dynamics in perceptual tasks. In *International Conference on Robotics and Systems*, 2005.

[6] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *IEEE International Conference on Image Processing*, 2002.

[7] LifeAlert. http://www.lifealert.com.

[8] D. G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, 1999.

[9] D. G. Lowe. Distinctive image features from scale invariant keypoints. *International Journal of Computer Vision*, 2004.

[10] H. Nait-Charif and S. J. McKenna. Activity summarisation and fall detection in a supportive home environment. In *International Conference on Pattern Recognition*, 2004.

[11] S. Ou, D. R. Karuppiah, A. Fagg, and E. Riseman. An augmented virtual reality interface for assistive monitoring of smart spaces. In *IEEE International Conference on Pervasive Computing*, 2004.

[12] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[13] P. Roessler, S. Consolvo, and B. Shelton. Phase 2 of Computer-Supported Coordinated Care project. Technical Report IRS-TR-04-006, Intel Research Seattle, 2004.

[14] B. U. Toreyin, Y. Dedeoglu, and A. E. Cetin. Hmm based falling person detection using both audio and video. In *IEEE Workshop on Human-Computer Interaction*, 2005.

[15] Tunstall Fall Detector. http://www.tunstallaustralasia.com/fall_detector.php.

[16] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.