Sensors and Materials, Vol. 37, No. 1 (2025) 117–134
MYU Tokyo

S & M 3895

Embedded Bed-exit Monitoring System Using Deep Learning

Chi-Huang Shih[1] and Yeong-Yuh Xu[2*]

[1]Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 41170, Taiwan
[2]Department of Artificial Intelligence and Computer Engineering, National Chin-Yi University of Technology, No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 41170, Taiwan

(Received August 22, 2024; accepted December 27, 2024)

Keywords: bed-exit monitoring, deep learning, bed-exit behavior recognition, fall hazard minimization

In this study, we present a practical bed-exit monitoring system designed specifically for healthcare settings, distinguished by its focus on privacy, accuracy, cost, and ease of use. The NVIDIA Jetson Xavier, a compact System-on-Module, powered the system, using a camera serial interface for video data acquisition. Specifically, the presented system aims to recognize the bed-exit behavior from a series of images with narrow fields of view for privacy preservation. Our approach encompasses a three-stage process for detecting, tracking, and classifying human body trunk movements relative to a bed. First, the You Only Look Once (YOLO) algorithm detects the human body trunk within the scene. Following detection, the Simple Online and Real-time Tracking (SORT) algorithm tracks the detected body trunk objects across frames. Finally, deep learning techniques such as long short-term memory (LSTM) or gated recurrent unit (GRU) networks classify the tracked objects' actions into three categories: getting off the bed, being on the bed, and returning to the bed. Our experimental findings demonstrate that this system achieves a high accuracy rate of 97.97% and operates at a processing speed of 7.1 frames per second. This methodology offers precise bed-exit monitoring and represents a significant step forward in improving patient safety and the efficiency of care in healthcare environments, underscoring its importance in minimizing fall hazards and enhancing patient care quality.

1. Introduction

According to statistics from the World Health Organization on individuals over 65, about 30 to 50% of patients/residents in care facilities experience falls annually, with 40% of those who fall experiencing recurrent falls.(1) The widespread concern over falls stems from their potential to cause significant subsequent injuries, including head injuries or even death.(2) Data from the Centers for Disease Control and Prevention in 2015 indicate that the medical costs incurred from falls among Americans aged 65 and older amount to approximately $50 billion annually, with the costs related to severe injuries being $754 million.(3,4) In addition to medical expenses, other incalculable social costs include long-term care expenses due to fall-induced disabilities, work, time losses, and decreases in quality of life.

*Corresponding author: e-mail: yyxu@ncut.edu.tw
https://doi.org/10.18494/SAM5325

ISSN 0914-4935

For patients at high risk of falling, detecting bed-exit behaviors is considered the first defense against falls. Bed-exit behavior recognition is a branch of Human Activity Recognition. With advancements in sensing technology, the market has seen a diversification in bed-exit detection products. These technologies can be broadly categorized into image-based and non-image-based types. Non-image-based sensing sources can be divided into stationary and wearable types, including IR rays that detect distance and obstruction,[5–7] pressure sensors that detect weight and pressure,[8] and piezoelectric devices that sense object deformation.[9] Wearable sensors typically include accelerometers and gyroscopes that detect the wearer's motion.[10,11]

Image-based sensing sources include two-dimensional RGB color images,[12,13] thermal images,[14–16] and three-dimensional light detection and ranging.[17] The process generally involves capturing human images with recording devices, identifying human features (body, head, and limbs) from these images to obtain movement information, and feeding this information into a deep learning network for training, testing, and validation to classify bed-exit behaviors. Popular deep learning networks include convolutional neural networks (CNN)[18] and long short-term memory (LSTM).[19] Researchers such as Chen *et al.* have used deep imaging to create datasets with images sized 32 × 24 (width × length), categorizing behaviors into lying in bed, getting up from bed, and rolling out of bed; these behaviors are processed through two layers of CNN networks followed by a max-pooling layer, and finally classified by a fully connected layer.[12] Inoue *et al.* used a color camera to capture human figures and then employed the pose estimation technique OpenPose to calculate and generate skeleton information describing human postures.[13] On the basis of the relative positions of the skeleton nodes and the bed, we established a bed-exit behavior recognition system centered on LSTM. The LSTM network output consists of two categories: bed-exit and other behaviors. Of 3084 data entries and videos, 308 (approximately 10% of the dataset) were used as validation data. This method achieved a high classification rate of 99.2%. Chiu *et al.* opted for thermal imaging technology, which is less invasive to privacy, capturing ten frames per second of 1 × 64 temperature data, and classified behaviors using CNN into three categories: bed-exit, lying in bed sleeping, and other in-bed behaviors, achieving an average discrimination accuracy of 92.77% across 360 video datasets, with bed-exit behaviors accurately identified 99.23% of the time.[14]

In general, image-based technologies offer better accuracy and convenience but lower privacy and higher costs. In contrast, non-image-based technologies have the advantage of lower privacy concerns and costs. Still, they may suffer from higher false positive rates owing to interference from various activities of caregivers and patients. In this study, we focus on using image-based sensing for bed-exit detection, employing narrow-field-of-view (NFV) images to cover only a specific part of a broad space (e.g., the bed) to minimize the capture of spatial information of the patient and avoid privacy issues involving other individuals. Utilizing NFV images as the sensing source, we developed a portable, embedded system for bed-exit behavior detection, comprising four stages: object detection, object tracking, data segmentation, and behavior recognition. The first two stages are aimed at detecting the presence of a human body and capturing its movement coordinates within the detection space. In the object detection stage, the body's torso is identified to exclude caregiver limb movements in NFV images. Subsequently, the spatial coordinate changes generated by human movement are used for behavior recognition

through deep learning networks. Since the spatial coordinates observed over time show continuous changes corresponding to human behaviors or actions, these changes are treated as time-series data. In the data segmentation stage, we employed a sliding window technique to segment the time-series data into segments containing historical and recent data. In the behavior recognition stage, the deep learning network calculates the behavior category using the input data segments, referring to previously trained parameters. To achieve better performance, we explored the effects of window size and step size in the sliding window technique. We compared the behavior recognition performance of several classic deep learning networks, including CNN, LSTM, gate recurrent unit (GRU),[20] and their fusion versions. The training process of the deep learning networks utilized the Taguchi method to identify the optimal performance parameters.[21,22] Experimental results showed the following: (1) LSTM achieved the highest behavior recognition accuracy of 97.97% with an overall system running speed of 7.1 frames per second, (2) time-domain networks such as LSTM and GRU showed slightly higher accuracy than did spatial-domain CNN, and (3) fusion of CNN with time-domain networks yielded slightly higher accuracy and speed than CNN alone but slightly lower than that of time-domain networks.

The structure of this paper is as follows. In Sect. 2, we describe the system architecture of the bed-exit behavior recognition system and the technical details of the four stages. In Sect. 3, we present the experimental design and performance comparison. Finally, we conclude and discuss future work in Sect. 4.

2. Materials and Methods

In this section, we outline the methods and technologies used to develop and evaluate our bed-exit behavior monitoring system. We aim to achieve accurate and real-time monitoring of bedridden individuals by integrating advanced object detection and tracking algorithms. Our approach leverages lightweight models and efficient tracking techniques to ensure that the system is effective and suitable for deployment in real-world healthcare environments. Below, we detail the specific methodologies and tools employed in our study.

2.1 System overview

Figure 1 illustrates scenarios of tracking objects using different fields of view. Compared with a wide field of view (WFV) covering a larger area, NFV only includes a specific location, such as a bed. Generally, the smaller coverage area of NFV limits the number of detectable objects and recognizable behaviors, making NFV more suitable for monitoring simple behaviors within a specific location. On the other hand, NFV reduces the tracking of nontarget objects, thereby maintaining higher privacy. To align with real-world caregiving environments, we utilized RGB cameras with infrared night vision capabilities deployed at the head or foot of the bed, capturing NFV images during both day and night to monitor bed-exit behaviors. As shown in Fig. 1, installing the camera at the head of the bed is referred to as front-end deployment, while installation at the foot of the bed is referred to as rear-end deployment. Front-end

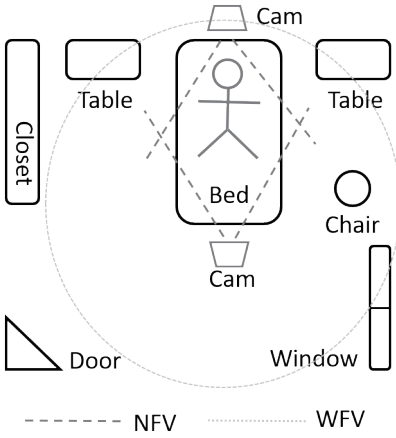

Fig. 1. Object tracking range in the ward/room. Cam stands for the camera.

deployment primarily captures the back and side of the bedridden person but is more likely to capture the faces of other individuals. Rear-end deployment captures more of the bedridden person's front and less of others. Figure 2 shows NFV images captured during the day and night using rear-end deployment. Owing to the close distance and narrow field, NFV images generally capture the torso and head of the bedridden person, with other individuals being only partially visible, such as hands or incomplete heads and bodies.

Figure 3 shows the bed-exit behavior recognition system developed in this study using NFV images. This system consists of four main modules: object detection, object tracking, data segmentation, and behavior recognition. The object detection module aims to detect human bodies, providing spatial information of detected bodies in the image {object length, object width, center X coordinate, center Y coordinate} = {L, W, X, Y}. The object tracking module distinguishes the bedridden person from others in the image. Additionally, owing to the limited field of view, movements in NFV images exhibit significant coordinate fluctuations. The object tracking module provides smoother object movement trajectories: {object ID, center X coordinate, center Y coordinate} = {ID, X, Y}. Each frame's spatial information is linked in the data segmentation module and segmented into discrete time-series data. A segment of time-series data is referred to as a window. The behavior recognition module uses deep learning networks to quickly expand behavior recognition types to classify bed-exit behaviors based on these data windows. The data segmentation and behavior recognition modules require appropriate parameter selection to optimize the system's performance. Given the numerous hyperparameters in deep learning networks, the behavior recognition module uses the Taguchi method for optimal parameter selection.

In the Taguchi method, less quality loss indicates higher quality. Let n represent the total number of products being evaluated, y_i the characteristic of the output product, and m the target value for the quality. *MSD* refers to the mean squared deviation of the product sample, defined as

$$MSD = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - m\right)^2. \tag{1}$$

(a) (b)

Fig. 2. (Color online) NFV images captured from the rear-end camera: (a) day and (b) night.

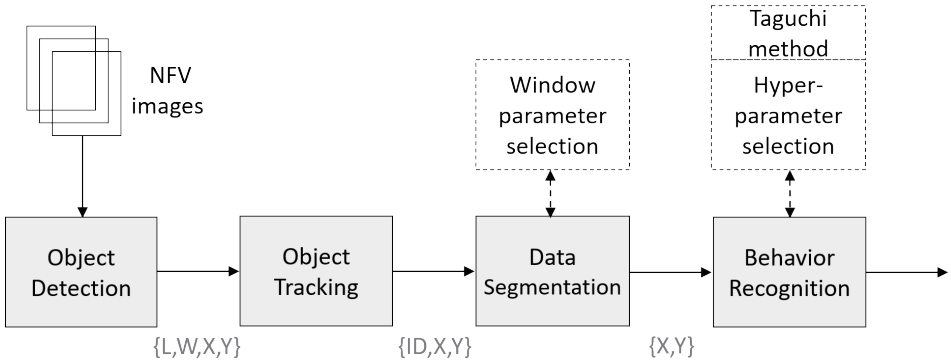

Fig. 3. Bed-exit behavior recognition system architecture.

The total quality loss function can be categorized into three types: larger-the-better (LTB), smaller-the-better (STB), and nominal-the-best (NTB). In this study, the LTB criterion is used as the reference for selecting the optimal parameters. In this case, the value of m in Eq. (1) is set to 0:

$$\frac{1}{n}\sum_{k=1}^{n}k\left(\frac{1}{y_i}-m\right)^2=\frac{1}{n}\sum_{k=1}^{n}k\left(\frac{1}{y_i}\right)^2. \tag{2}$$

The Taguchi method uses the signal-to-noise ratio (*SNR*) to compare the effects of different control factors and noise factors on the final result within the same experiment. The *SNR* for the LTB criterion is expressed as

$$\eta=10\log\left[\frac{1}{n}\sum_{k=1}^{n}\left(\frac{1}{y_i}\right)^2\right], \tag{3}$$

where η refers to the calculation group for each factor level in the orthogonal array. For example, if a control factor has two levels, it is necessary to calculate the *SNR* for level 1, denoted as η_1, and for level 2, denoted as η_2. A larger *SNR* indicates a smaller quality loss for that level, making it the more ideal level.

2.2 Object detection

Traditional object detection methods can be categorized into two types: two-stage and one-stage. In two-stage methods, common approaches include fast R-CNN[23] and faster R-CNN.[24] Although two-stage methods can enhance accuracy, the more regions generated by the feature pyramid networks (FPNs), the greater the processing load for regions of interest, leading to slower detection speeds. One-stage methods achieve object detection by directly predicting object classes and bounding boxes using FPN to address this efficiency issue. Among one-stage methods, the single shot multibox detector[25] and the You Only Look Once (YOLO) series[26] are notable examples. One-stage methods offer the advantage of fast operation, making them suitable for real-time object detection. These methods provide quick and accurate object detection, meeting real-time requirements, although they may not perform as well in multi-object and complex feature classification scenarios.

In this study, we used the YOLOv4-Tiny object detection method. YOLOv4-Tiny is a lightweight model that detects and locates multiple object classes within images.[27] Compared with the standard YOLOv4, YOLOv4-Tiny is even more lightweight, with a smaller model size and faster inference speed. Owing to its compact size and quick inference, it is suitable for deployment on embedded devices. As shown in Fig. 2, we use human detection as the basis for identifying bed-exit behaviors. To avoid confusion between background objects and humans, YOLOv4-Tiny labels humans by including the torso and upper limb regions (green frame in Fig. 2).

2.3 Object tracking and data segmentation

In this study, the object-tracking method is Simple Online and Real-time Tracking (SORT).[28] SORT is a multi-object tracking method that employs a Kalman Filter[29] and the Hungarian Algorithm[30] to track and identify targets in images. It maintains accuracy even when targets appear, disappear, or occlude each other, effectively operating in complex environments. Figure 4 shows the tracking results of six bed-exit behaviors of the bedridden person. In Fig. 4(a), the bedridden person exits the bed from the right side of the image. A significant change in the Y-axis value can be observed during the bed exit, with the *X*-axis value gradually increasing, indicating that the target is moving to the right. Figure 4(c) shows a similar observation for bed exit from the left side, but the *X*-axis value gradually decreases as the target moves to the left. Figure 4(b) illustrates the bedridden person returning to bed from the right side of the image. The *X*-axis coordinate decreases, indicating that the target moves from right to left. When the target reaches and lies back on the bed, the *Y*-axis coordinate shows a horizontal displacement, and the *X*-axis coordinate gradually stabilizes. Figure 4(d) depicts the bedridden person

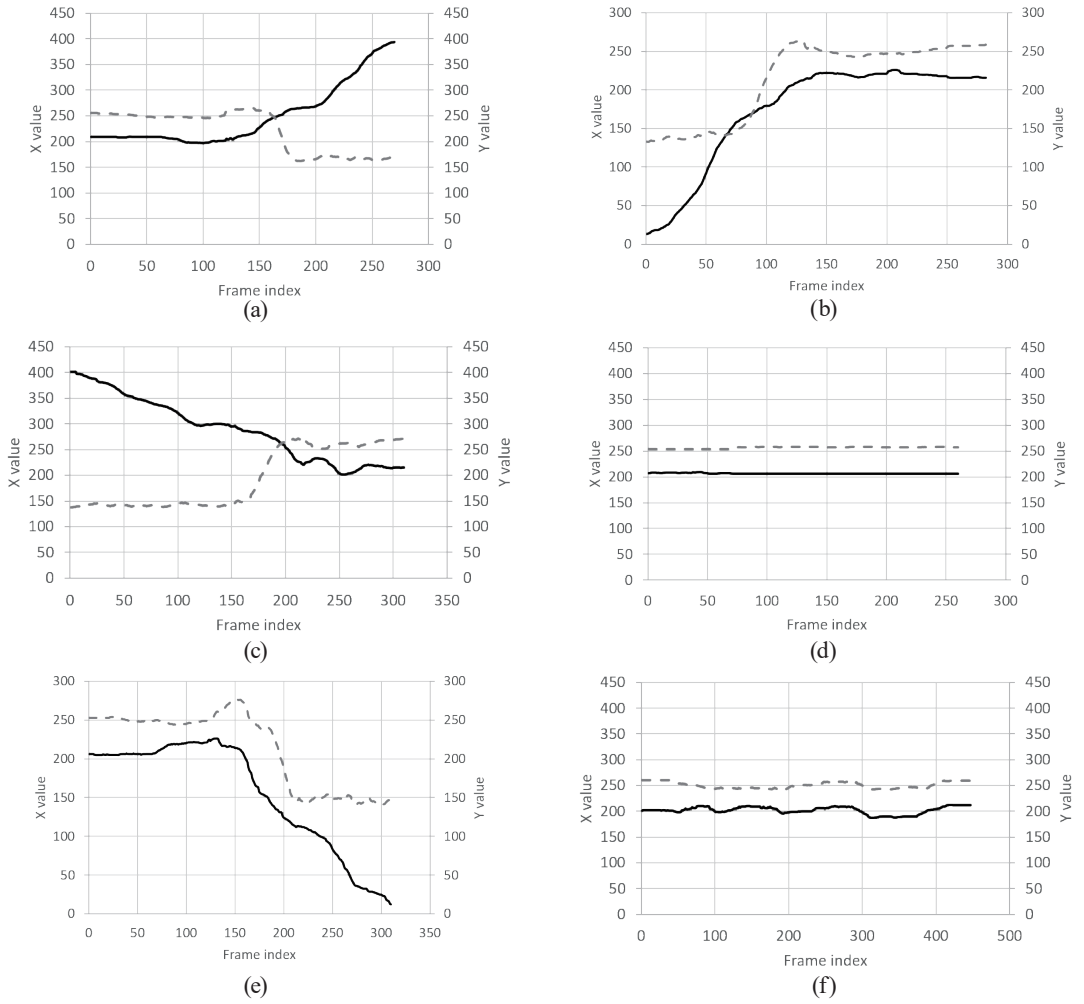

Fig. 4. Coordinate traces for bed-exit behaviors: (a) right exit, (b) right return, (c) left exit, (d) left return, (e) lie still, and (f) turn over. The solid black line represents the *X*-axis trace, and the dotted grey line represents the *Y*-axis trace.

returning to bed from the left side, showing similar observations as in Fig. 4(b). Nevertheless, the *X*-axis coordinate increases, representing movement from left to right until it stabilizes upon returning to bed. Figure 4(e) shows the coordinate trajectory of the bedridden person maintaining a supine/prone position without movement. Figure 4(f) shows the coordinate trajectory of the bedridden person turning over in bed. Compared with Fig. 4(e), the *X*-axis and *Y*-axis coordinates display slight oscillations.

After performing object tracking, segmenting the data for subsequent action recognition is essential. In this study, we use the sliding window technique for data segmentation. This method involves moving a fixed-size window over the sequential data, to capture overlapping segments that serve as input for the action recognizer.

The sliding window technique is critical in dividing continuous tracking data into manageable segments. Each segment, or "window", contains a snapshot of the tracked coordinates and movements within a specific time frame. This approach ensures that the action recognizer can

analyze smaller, consistent data portions, making it easier to identify distinct behaviors accurately.

Several considerations must be taken into account when using the sliding window technique. The size of the sliding window must be carefully chosen. A window that is too small might miss crucial contextual information, whereas a very large window can include multiple actions, leading to confusion in recognition. The degree of overlap between consecutive windows is vital as well. Sufficient overlap ensures that transitional movements between actions are captured, which is critical for accurate behavior recognition. Maintaining data continuity is essential, and the sliding window should move incrementally across the dataset to ensure no relevant information is lost between segments. While more oversized windows and higher overlap can improve accuracy, they also increase computational load. Therefore, balancing accuracy with efficiency is critical to optimizing system performance.

Using the sliding window method, we ensure that the behavior recognition system receives well-segmented, contextually rich data, enhancing the accuracy and reliability of action identification in our bed-exit monitoring system.

2.4 Behavior recognition

In this study, we recognize five bed-exit behaviors: exiting the bed from the right side, exiting from the left side, returning to the bed from the right side, returning from the left side, and lying in bed. We utilized CNNs, LSTM networks, and GRU networks, alongside hybrid models combining CNN with LSTM and CNN with GRU, to recognize these behaviors and compare their performances.

2.4.1 CNN model

CNNs are widely used in deep learning for their ability to extract local features through convolutional layers and reduce dimensionality via pooling layers, effectively capturing spatial and temporal relationships in data. The CNN architecture includes 1D convolutional layers, max-pooling layers, flattening layers, and fully connected layers. The input layer receives a 2D time series $X \in R^{T\times 2}$ with a time step T. This input data is then processed by a 1D convolutional layer, which extracts local features by sliding filters over the input data. A 1D convolutional layer processes the data to extract local features, represented as

$$Y^{(1)} = f\left(w^{(1)} * X + b^{(1)}\right), \tag{4}$$

where $w^{(1)}$ is the convolution kernel, $b^{(1)}$ is the bias, and f is the ReLU activation function. The output is then passed to a max-pooling layer with a pool size of two to reduce dimensionality and computational load, followed by a flattening layer to convert the output into a 1D vector for the fully connected layers. The purpose of the max-pooling operation is to reduce the dimensionality of the feature map, thereby decreasing the computational load and mitigating the risk of

overfitting. The output from the pooling layer then passes through a flattening layer, converting the multidimensional output into a one-dimensional vector suitable for the fully connected layers.

The first fully connected layer (dense) contains 64 neurons,

$$Y^{(3)} = f\left(W^{(2)}Y^{(2)} + b^{(2)}\right), \tag{5}$$

where $W^{(2)}$ is the weight matrix of the fully connected layer, $b^{(2)}$ is the bias, and $Y^{(2)}$ is the output of the flattening layer. The second fully connected layer (output layer) contains five neurons, corresponding to the five bed-exit behavior categories:

$$Y^{(4)} = \text{softmax}\left(W^{(3)}Y^{(3)} + b^{(3)}\right). \tag{6}$$

The softmax function converts the output into a probability distribution. The network, trained using backpropagation and the cross-entropy loss function,

$$L = -\sum_{i=1}^{5} y_i \log\left(\hat{y}_i\right), \tag{7}$$

where y_i is the actual label and $\hat{y}_i$ is the predicted probability, can accurately recognize the input time series category.

2.4.2 LSTM model

LSTM networks are a type of recurrent neural network (RNN) suitable for time-based data. They capture long-term dependences and avoid the vanishing gradient problem. LSTM introduces memory cells and gates to selectively retain or discard information, maintaining stable memory over time. The LSTM architecture includes the LSTM layers and fully connected layers. Figure 5 shows the architecture of an LSTM unit.

Each LSTM unit contains a forget gate, an input gate, and an output gate. The input layer receives a 2D time series $X \in R^{T\times 2}$. The LSTM layer, with hidden units, processes the data in the following steps.

1) Forget gate:

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right). \tag{8}$$

2) Input gate:

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right), \tag{9}$$

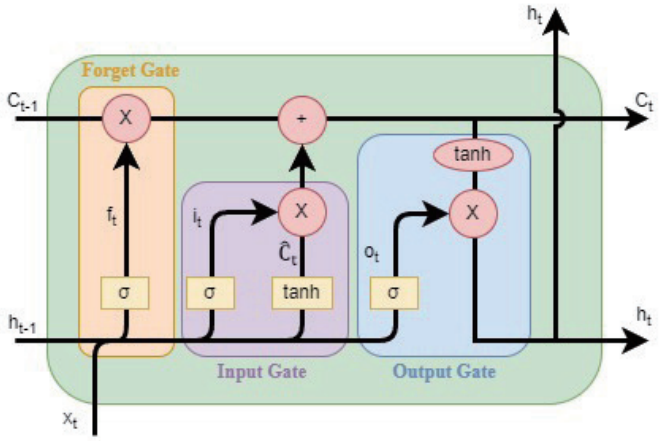

Fig. 5. (Color online) Architecture of an LSTM unit.

$$\hat{C}_t = \tanh\left(W_C \cdot \left[h_{t-1}, x_t\right] + b_C\right). \tag{10}$$

3) Cell state update:

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t. \tag{11}$$

4) Output gate:

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right). \tag{12}$$

5) Hidden state:

$$h_t = o_t * \tanh\left(C_t\right). \tag{13}$$

Here, σ is the sigmoid activation function, tanh is the hyperbolic tangent activation function, W_f, W_i, W_C, and W_o are the weight matrices for the forget gate, input gate, cell state, and output gate, respectively, b_f, b_i, b_C, and b_o are the corresponding biases, h_t is the hidden state at the current time step, and C_t is the cell state at the current time step. The final hidden state h_t is passed to the fully connected layer:

$$Y = \text{softmax}\left(W_d \cdot h_T + b_d\right). \tag{14}$$

The softmax function outputs a probability distribution for the five categories. The network, trained using backpropagation and the cross-entropy loss function, can accurately recognize the input time series category.

2.4.3 GRU model

GRU is a simplified version of LSTM. It combines input and forget gates into a single update gate and uses a reset gate for efficiency. The GRU architecture includes GRU layers and fully connected layers. Figure 6 shows the architecture of a GRU unit.

The input layer receives a 2D time series $X \in R^{T\times 2}$. The GRU layer processes the data in the following steps.

1) Update gate:

$$z_t = \sigma\left(W_z \cdot \left[h_{t-1}, x_t\right] + b_z\right). \tag{15}$$

2) Reset gate:

$$r_t = \sigma\left(W_r \cdot \left[h_{t-1}, x_t\right] + b_r\right). \tag{16}$$

3) Candidate hidden state:

$$\hat{h}_t = \tanh\left(W \cdot \left[r_t * h_{t-1}, x_t\right] + b\right). \tag{17}$$

4) Hidden state update:

$$h_t = \left(1 - z_t\right) * h_{t-1} + z_t * \hat{h}_t. \tag{18}$$

Here, σ is the sigmoid activation function, tanh is the hyperbolic tangent activation function, W_z, W_r, and W are the weight matrices for the update gate, reset gate, and candidate hidden state, respectively, b_z, b_r, and b are the corresponding biases, and h_t is the hidden state at the current time step. The final hidden state h_t is passed to the fully connected layer:

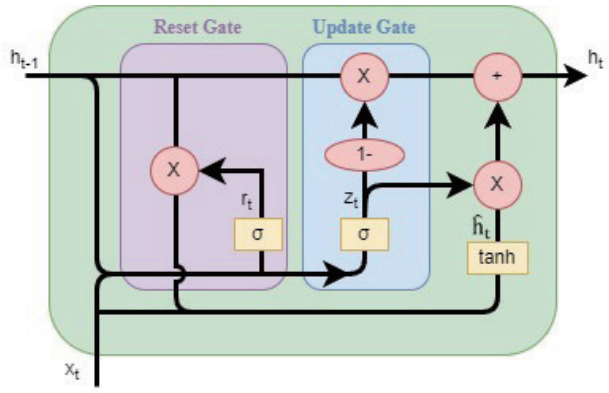

Fig. 6. (Color online) Architecture of a GRU unit.

$$Y = \text{softmax}\left(W_d \cdot h_T + b_d\right). \tag{19}$$

The softmax function outputs a probability distribution for the five categories. The network, trained using backpropagation and the cross-entropy loss function, can accurately recognize the input time series category.

2.4.2 Hybrid model

Hybrid models combine the ability of CNNs to extract local features and the ability of RNNs to model temporal dependences, making them advantageous for time series data classification. In this study, we designed hybrid models combining CNN with LSTM and CNN with GRU. When designing a hybrid model that combines CNN and LSTM/GRU, the trade-off between complexity and performance needs to be dynamically adjusted in accordance with the application's requirements. For scenarios that demand high accuracy, it is appropriate to increase the complexity of the model. In contrast, for situations that require rapid response, lightweight models should be prioritized. In the application of Embedded Bed-Exit Monitoring, real-time prediction is crucial. A model with fewer parameters and lower computational effort is more suitable for running on embedded systems. Therefore, when determining the optimal architecture and parameters for the hybrid model, evaluating the performance of different architectures and parameter combinations is essential, ensuring that the model can provide predictions within a reasonable time frame without sacrificing accuracy.

The hybrid model extracts local spatiotemporal features through convolutional and pooling layers, captures global temporal dependences through LSTM/GRU layers, and classifies data through fully connected layers. The input layer receives a 2D time series $X \in R^{T\times 2}$. After convolution and pooling, the flattened output is passed to the RepeatVector layer, the LSTM/GRU layer, and finally, to the fully connected layers. The network, trained using backpropagation and the cross-entropy loss function, can accurately recognize the input time series category.

3. Results

3.1 Object detection results

We established an NFV image dataset for bed-exit behaviors, including bed exits, lying in bed, and returning to bed. Table 1 lists the dataset statistics used in the object detection stage. To align with real-world care settings, the dataset's images are primarily divided into daytime and nighttime categories. For each image mode, the data ratio for training, validation, and testing is 7:2:1, with the test data being independent of training and validation purposes. The total amount of data in the dataset is 6937 images. As stated in Sect. 2.2, we labeled the torso, including the upper limbs, to identify humans in NFV images. On the basis of the detection results, we can obtain the following classification parameters: true positive (*TP*), false positive (*FP*), false negative (*FN*), and true negative (*TN*). With these parameters, the main object detection performance metrics are precision, recall, and *F*1 score (*F*1) in order:

Table 1
Statistics in the dataset.

Mode	Train	Validate	Test
Day	2983	658	329
Night	2017	634	316
All	5000	1292	645

$$Precision = \frac{TP}{TP + FP}, \tag{20}$$

$$Recall = \frac{TP}{TP + FN}, \tag{21}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}. \tag{22}$$

Furthermore, the area under the precision and recall curve is the average precision (*AP*). Table 2 presents the validation and test results of human detection using Yolov4-Tiny in this study. In the validation results, the precision and AP for nighttime images were slightly lower than those for daytime images: daytime images achieved a precision of 99% and an AP of 99.68%, while nighttime images had a precision of 98% and an AP of 97.63%. A similar observation can also be seen in the test results. Overall, both validation and test results showed that we could achieve a precision of 99% in daytime images, whereas the precision for nighttime images was above 98%. These results indicate that effective human detection obtained during the object detection stage can be utilized for subsequent behavior recognition.

3.2 Data segmentation and behavior recognition results

To observe the impact of sliding window parameters, we conducted a series of experiments using an LSTM as the deep learning network model for behavior recognition and compared the performance differences of different parameter settings within the same network model. In the experiments, the deep learning network utilized a single layer of LSTM, followed by a dropout layer, a dense layer, and an output layer sequentially; the hyperparameters for neurons and dense layer units were fixed at 100, with a dropout parameter of 0.5. The activation function for the dense layer was ReLU, while the output layer used softmax. The window size was set to {50, 100, 200}, and the step size was set to {5, 10, 20, 40}. The accuracy of behavior recognition is defined as follows:

$$Accuracy = \frac{TP + TN}{\left(TN + TP + FN + FP\right)}. \tag{23}$$

Table 2
Performance results in the object detection stage.

	Validate				Test			
Mode	Precision (%)	Recall (%)	*F*1 score (%)	*AP* (%)	Precision (%)	Recall (%)	*F*1 score (%)	*AP* (%)
Day	99	99	99	99.68	99	99	99	99.17
Night	98	95	96	97.36	99	96	98	97.98
All	99	97	98	98.57	99	98	98	99.30

Table 3 illustrates the bed-exit behavior recognition results for various window and step size combinations. Besides accuracy, we also recorded the timing of bed-exit behavior occurrences for each parameter combination. From Table 3, we can observe that (i) when the window size is fixed at 50, increasing the step size results in lower accuracy and delays the recognition timing of bed-exit behavior; (ii) when the window size is increased to 100 or 200, the impact of step size on performance is not significant; and finally, (iii) the optimal sliding window parameters {window size, step size} are {100, 10} and {200, 20}. Since a smaller window size allows for quicker recognition of the target's behavior, the parameter combination of {100, 10} is preferred in this study.

In the behavior recognition stage, we used various deep learning networks to compare the performance of behavior classification and selected the hyperparameters for each network using the Taguchi method. Table 4 shows the hyperparameter selection results for five deep-learning network models. We used 159 bed-exit behavior videos to evaluate the identification performance of three behaviors: in-bed, bed-exit, and return-to-bed. For in-bed behavior, there were 22 daytime videos and 15 nighttime videos; for bed-exit behavior, there were 31 daytime videos and 28 nighttime videos; for return-to-bed behavior, there were 34 daytime videos and 29 nighttime videos. With the hyperparameters shown in Table 4, the performances the five behavior recognition networks were compared and the results are summarized in Table 5. Besides accuracy and loss, we also calculated the memory space required to run the network—the number of hyperparameters and the floating-point operations per second (FLOPs) were related to execution speed. The FLOPs varies depending on the operations of the deep learning model. The FLOPs calculation for a CNN is

$$FLOPs_{CNN} = \left(2 \times C_{in} \times K_h \times K_w - 1\right) \times H \times W \times C_{out}, \tag{24}$$

where C_{in} is the number of input channels, $K_h \times K_w$ is the kernel size, H and W are the height and width of the output feature map, and C_{out} is the number of output channels. The FLOPs calculation for an LSTM is

$$FLOPs_{LSTM} = \left(E + H\right) \times H \times 4 \times 2, \tag{25}$$

where E is the embedding dimension, which represents the dimensionality of word vectors, H is the hidden state dimension, representing the number of neurons in the LSTM, and 4 represents the four nonlinear transformation blocks in an LSTM. FLOPs for a GRU is defined as

Table 3
Accuracy and time results with various window and step sizes.

Step size	Window size					
	50		100		200	
	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)
5	87	31.29	97	30.54	100	32.75
10	80	33.47	100	32.02	97	32.02
20	70	33.47	90	34.93	100	32.02
40	43	36.39	97	32.02	97	34.93

Table 4
Selected hyperparameters for five deep-learning network models.

Model	Hyperparameters
CNN	Filter = 16, Kernel size = 3, Dense = 64
LSTM	Neurons = 128, Dense = 64
GRU	Neurons = 128, Dense = 64
CNN + LSTM	Filter = 16, Kernel size = 5, Neurons = 64, Dense = 16
CNN + GRU	Filter = 8, Kernel size = 5, Neurons = 64, Dense = 8

Table 5
Performances of deep-learning network models.

Model	Accuracy (%)	Parameters	FLOPs
CNN	96.87	25077	30719
LSTM	97.97	477061	134400
GRU	97.89	59269	102368
CNN + LSTM	97.39	112149	297727
CNN + GRU	97.25	48653	79279

$$FLOPs_{GRU} = \left(E + H\right) \times H \times 3 \times 2. \tag{26}$$

In Eq. (23), 3 represents the three nonlinear transformation blocks in the GRU, which is one less gating mechanism compared with the LSTM. For hybrid models, such as CNN+LSTM and CNN + GRU, the FLOPs result is the sum of the FLOPs of the two models:

$$FLOPs_{CNNLSTM} = FLOPs_{CNN} + FLOPs_{LSTM}, \tag{27}$$

$$FLOPs_{CNNGRU} = FLOPs_{CNN} + FLOPs_{GRU}. \tag{28}$$

Table 5 reveals that under the condition of selecting hyperparameters based on accuracy, all network models can achieve an accuracy between 97 and 98%, with LSTM performing the best, reaching an accuracy of 97.97%, and GRU having the next best performance with 97.89% accuracy. For the three single networks (i.e., CNN, LSTM, and GRU), CNN has a slightly lower accuracy than LSTM and GRU but has fewer parameters and computational FLOPs. On the other hand, for the two types of hybrid network model, CNN + GRU can achieve slightly lower

Table 6
Behavior recognition accuracy results.

Mode	On bed	Off bed	Return
Day	100% (22/22)	100% (31/31)	100% (34/34)
Night	100% (15/15)	100% (28/28)	100% (29/29)
All	100% (37/37)	100% (59/59)	100% (63/63)

accuracy with fewer parameters and computational effort. In summary, fewer parameters and computational effort are more suitable for running on embedded systems. Considering the higher loss of the single CNN model, we adopted the hybrid CNN + GRU model for the behavior recognition network.

Referencing the description of behavior recognition outputs in Sect. 2.4, we show the results in Table 5 on a window-by-window basis. In contrast, Table 6 shows the final behavior recognition results based on a majority vote. The daytime and nighttime video test results for three different behaviors indicate that the bed-exit behavior recognition system proposed in this study can achieve a 100% recognition rate, thus avoiding unnecessary care burdens caused by false reports.

4. Conclusions

In this study, we presented an embedded bed-exit monitoring system with a combination of the Jetson Xavier platform and camera serial interface cameras. The proposed system applies YOLO for object detection, SORT for object tracking, and deep learning models such as LSTM/GRU for bed-exit behavior recognition. The experimental results showed that the proposed system has excellent recognition accuracy.

To protect patient privacy, the system employs NFV images to monitor bed-exit behaviors. By capturing only the specific area of the patient's bed for recognition, the capture of irrelevant images by the system is reduced. This approach ensures that the monitoring process does not infringe on the privacy of others in the surrounding area.

We chose YOLOv4-Tiny for object detection owing to its light weight and high processing efficiency, which are suitable for embedded systems. The experimental results showed that the system achieves high accuracy and recall rates day and night, demonstrating its stability under different lighting conditions. This is important for medical equipment used in continuous monitoring, as lighting conditions can vary significantly over time. Using the SORT algorithm for object tracking enhances the system's ability to accurately track the human torso, even when the target is occluded or there are multiple moving objects. This is crucial in dynamic environments such medical facilities, where the frequent movement of patients and caregivers increases tracking complexity.

The five deep learning models used in behavior recognition performed well in recognizing bed-exit behaviors. Among the three single-network models, LSTM performed the best. Between the two hybrid network models, CNN + GRU achieved slightly lower accuracy with fewer parameters and computational requirements. Considering that lower parameter counts and computational loads are more suitable for embedded system operation, we adopted the CNN + GRU hybrid model for behavior recognition.

In summary, the embedded bed-exit monitoring system developed in this study effectively monitors patient movements in healthcare environments. The system integrates deep-learning-based techniques for object detection, tracking, and behavior recognition, and shows high recognition accuracy and real-time performance while protecting patient privacy. The main achievements of this study are as the follows:

1) Efficient use of NFV images to achieve precise monitoring while protecting privacy.
2) Utilizing the YOLOv4-Tiny model to achieve high accuracy and recall rates, suitable for real-time applications in embedded devices.
3) Implementing the SORT algorithm for reliable object tracking, and capability to handle dynamic and complex environments.
4) Adoption of a CNN + GRU hybrid network model suitable for embedded system operation, achieving excellent behavior recognition performance with lower parameter count and computational load, and
5) Achieving a processing speed of 7.1 frames per second, enabling real-time operation suitable for deployment in healthcare facilities.

Future research will enhance system capabilities by exploring more deep learning models, improving detection and tracking algorithms, and conducting extensive field tests in various healthcare environments to validate system performance and robustness. Additionally, integrating the monitoring system with existing healthcare information systems can provide a comprehensive patient care and management solution.

References

1 World Health Organization: WHO Global Report on Falls Prevention in Older Age (2008). https://www.who.int/publications/i/item/9789241563536
2 N. M. Peel, D. J. Kassulke, and R. J. McClure: Injury Prevention (2002) 280. https://doi.org/10.1136/ip.8.4.280
3 Centers for Disease Control and Prevention: Older Adult Falls Data (2024). https://www.cdc.gov/falls/data-research/index.html
4 C. S. Florence, G. Bergen, A. Atherly, E. R. Burns, J. A. Stevens, and C. Drake: J. Am. Geriatrics Soc. **66** (2018) 693. http://doi.org/10.1111/jgs.15304
5 H. Shibuya, S. Ogiwara, K. Ozawa, M. Kamishita, T. Uematu, and K. Inoue: J. Rural. Med. **57** (2008) 650. https://doi.org/10.2185/jjrm.57.650
6 R. Suzuki, S. Otake, T. Izutsu, M. Yoshida, and T. Iwaya: Telemed. e-Health **12** (2006) 146. http://doi.org/10.1089/tmj.2006.12.146
7 J. H. Shin, B. Lee, and K. Suk Park: IEEE Trans. Inf. Technol. Biomed. **15** (2011) 438. http://doi.org/10.1109/TITB.2011.2113352
8 C. Lu, J. Huang, Z. Lan, and Q. Wang: Proc. 2016 Int. Conf. Advanced Robotics and Mechatronics (2016) 59–64. https://doi.org/10.1109/ICARM.2016.7606895
9 H. Madokoro, K. Nakasho, N. Shimoi, H. Woo, and K. Sato: Sensors **20** (2020) 1415. https://doi.org/10.3390/s20051415
10 L. Atallah, B. Lo, R. King, and G.-Z. Yang: Proc. Int. Conf. Body Sensor Networks (IEEE, 2010) 24–29. https://doi.org/10.1109/BSN.2010.23
11 M. C. Vilas-Boas, M. V. Correia, S. R. Cunha, and P. Silva: Proc. IEEE 3rd Portuguese Meeting Bioengineering (IEEE, 2013) 1–6. https://doi.org/10.1109/EMBC.2013.6610607
12 T. Chen, R. Hsiao, C. Kao, W. Liao, and D. Lin: Proc. Int. Conf. Applied System Innovation (IEEE, 2017) 188–191. https://doi.org/10.1109/ICASI.2017.7988382
13 M. Inoue, R. Taguchi, and T. Umezaki: Proc. 2018 40th Annu. Int. Conf. IEEE Engineering in Medicine and Biology Society (IEEE, 2018) 5006–5009. https://doi.org/10.1109/EMBC.2018.8513460
14 S. Chiu, J. Hsieh, C. Hsu, and C. Chiu: Proc. 2018 Int. Conf. System Science and Engineering (ICSSE, 2018) 1–6. https://doi.org/10.1109/ICSSE.2018.8520032

15 M. Inoue, R. Taguchi, and T. Umezaki: Proc. 2019 41st Annu. Int. Conf. IEEE Engineering in Medicine and Biology Society (IEEE, 2019) 3208–3211. https://doi.org/10.1109/EMBC.2019.8857233
16 Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh: Proc. IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2017) 1302–1310. https://doi.org/10.1109/CVPR.2017.143
17 H. Miawarni, T. A. Sardjono, E. Setijadi, Wijayanti, D. Arraziqi, A. B. Gumelar, and M. H. Purnomo: Proc. 2020 Int. Conf. Computer Engineering, Network, and Intelligent Multimedia (2020) 1–5. https://doi.org/10.1109/CENIM51130.2020.9298000
18 Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner: Proc. IEEE (1998) 2278–2324. https://doi.org/10.1109/5.726791
19 S. Hochreiter and J. Schmidhuber: Neural Comput. **9** (1997) 1735. https://doi.org/10.1162/neco.1997.9.8.1735
20 K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio: Association for Computational Linguistics (2014) arXiv:1406.1078. https://doi.org/10.48550/arXiv.1406.1078
21 G. Taguchi: Proc. Int. Conf. Quality, Tokyo, Japan (1978).
22 J. L. Rosa, A. Robin, M. B. Silva, C. A. Baldan, and M. P. Peres: J. Mater. Process. Technol. **209** (2009) 1181. https://doi.org/10.1016/j.jmatprotec.2008.03.021
23 R. Girshick: Proc. 2015 IEEE Int. Conf. Computer Vision (2015) 1440–1448. https://doi.org/10.48550/arXiv.1504.08083
24 S. Ren, K. He, R. Girshick, and J. Sun: IEEE Trans. Pattern Anal. Mach. Intell. (2017) 1137. https://doi.org/10.1109/TPAMI.2016.2577031
25 W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed: Proc. Eur. Conf. Computer Vision (2016) 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
26 J. Redmon, S. Divvala, R. Girshick, and A. Farhadi: Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition (2016) 779–788. https://doi.org/10.1109/CVPR.2016.91
27 C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao: Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition (2021) 13024–13033. https://doi.org/10.1109/CVPR46437.2021.01283
28 A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft: Proc. 2016 IEEE Int. Conf. Image Processing (2016) 3464–3468. https://doi.org/10.1109/ICIP.2016.7533003
29 R. E. Kalman: J. Basic Eng. **82** (1960) 35. https://doi.org/10.1115/1.3662552
30 H. W. Kuhn: Naval Res. Logist. Q. **2** (1955) 83. https://doi.org/10.1002/nav.3800020109

About the Authors

Chi-Huang Shih received his Ph.D. degree in electrical engineering from National Cheng Kung University, Taiwan, R.O.C., in 2008. Currently, he is an associate professor of the Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung, Taiwan, R.O.C. His current research interests are machine learning, multimedia network communication, biomedical worn devices, embedded systems, and IoT applications. (chshih@ncut.edu.tw)

Yeong-Yuh Xu received his Ph.D. degree in computer science and information engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 2004. He is an assistant professor in the Department of Artificial Intelligence and Computer Engineering at National Chin-Yi University of Technology, Taichung, Taiwan. His research interests include pattern recognition, neural networks, machine learning, and content-based image/video retrieval. (yyxu@ncut.edu.tw)