Supplementary Information

Title:
An Eight-Camera Fall Detection System Using Human Fall Pattern Recognition via Machine Learning by a Low-Cost Android Box

Authors:
Francy Shu, MD1*, Jeff Shu, MS2

1Assistant Professor
Division of Neuromuscular Medicine
Department of Neurology
University of California, Los Angeles Medical Center
300 Medical Plaza, B200
Los Angeles, CA, USA 90095
626-616-8538
fshu@mednet.ucla.edu

2Chief Technology Officer
SpeedyAI, Inc.
19940 Ridge Estate Ct
Walnut, CA, 91789
626-674-1777
jeff@speedyai.com
Table of Contents

Histogram of Oriented Gradients (HOG) Feature Extraction ................................................................. 3
Relevance Vector Machine (RVM) Learning Framework ................................................................. 6
Testing Data .......................................................................................................................................... 7
Histogram of Oriented Gradients (HOG) Feature Extraction

Consider first a tiling of an image (a video frame) into a rectangular grid of cells, as shown in Supplementary Figure S1. Each cell is associated with it 25 bins whose values are initialized to 0. The purpose of these 25 bins will be addressed shortly. Spatially, each cell also consists of some number of pixels, and the insert in Figure S1 shows one such example in magnified view. For a pixel, we approximate its horizontal and vertical intensity value changes as $(\Delta x, \Delta y) = (R - L, T - B)$, where L, R, T, and B are the intensities of the left, right, top, and bottom spatial neighbors of the pixel, respectively (Supplementary Fig. S2). With this $(\Delta x, \Delta y)$, we compute a dot product with each of the 8 unit vectors shown in Supplementary Figure S3a representing 8 unsigned directions, and a dot product with each of the 16 unit vectors shown in Supplementary Figure S3b representing 16 signed directions, 8 of which are essentially the opposite of the 8 unsigned directions. Lastly, we note the direction corresponding to the unsigned case whose dot product has maximal magnitude, and the direction corresponding to the signed case whose dot product has maximal value. For ease of ongoing discussion, for the pixel under consideration, say the unsigned direction chosen is 67.5° and the signed direction chosen is 247.5°. Let $m_{\text{orientation}}$ be this maximal magnitude, which is the absolute value of the dot product with the 67.5° or the 247.5° unit vector. Finally, let $m_{\text{texture}}$ be the sum of the absolute values of all the 8 dot products from the unsigned directions. The choice of names “orientation” and “texture” will be discussed shortly.

![Image S1: Image tiling into cells.](#)

**Fig. S1** Image tiling into cells. Insert: Each cell consists of some number of pixels.

| T | Pixel | R |
|---|---|---|
| L |  |
| B |  |

![Image S2: The intensities of the left (L), right (R), top (T), and bottom (B) spatial neighbors used to calculate pixel intensity differential.](#)

**Fig. S2** The intensities of the left (L), right (R), top (T), and bottom (B) spatial neighbors used to calculate pixel intensity differential.

![Image S3: HOG orientation directions. a, Eight unit vectors representing eight unsigned directions. b, Sixteen unit vectors representing sixteen signed directions, eight of which are essentially the opposite of the eight unsigned directions.](#)

**Fig. S3** HOG orientation directions. a, Eight unit vectors representing eight unsigned directions. b, Sixteen unit vectors representing sixteen signed directions, eight of which are essentially the opposite of the eight unsigned directions.
Imagine for now that each of the 25 bins of a cell has a name. Eight of the names are unsigned 0°, unsigned 22.5°, ..., unsigned 157.5°. Sixteen of the names are signed 0°, signed 22.5°, ..., signed 337.5°. One of the names is texture. Then, for the cell C in which the pixel in the example above resides, we increment its bin values in the following manner.

\[
\text{bin[unsigned 67.5°]} \leftarrow \text{bin[unsigned 67.5°]} + m_{\text{orientation}} \\
\text{bin[signed 247.5°]} \leftarrow \text{bin[signed 247.5°]} + m_{\text{orientation}} \\
\text{bin[texture]} \leftarrow \text{bin[texture]} + m_{\text{texture}}
\]

Note that the unsigned and signed directions chosen in the manner above must always be identical or differ by 180°. The proof is straightforward and is hereby omitted. We traverse through the pixels in the image and compute the bin values as depicted above. Each cell’s bins will be incremented for multiple times as a cell typically consists of multiple pixels. Therefore, these bin values are not necessarily sparse.

An image typically consists of three channels representing its color information. In the HOG feature extraction algorithm above, it is not specified which channel the pixel intensity value comes from. There are two possibilities here. First, we carry out the algorithm three times, once for each color channel, and for each pixel simply collect our \(m_{\text{orientation}}\) and \(m_{\text{texture}}\) from whichever channel that would give the maximum value. Let us denote this possibility by Grayscale HOG, where each cell has 25 bins forming effectively a 25-vector. Second, we keep all the three outcomes and leave them intact, effectively expanding the bin size of each cell to \(25 \times 3 = 75\). Let us denote this possibility by Color HOG, where the bins of each cell are essentially a 75-vector.

It is worthwhile to pause here and consider the intuitive motivation of the above algorithm as reflected from the naming convention. We note that \(m_{\text{orientation}}\) is collected from the maximal dot product magnitude, which corresponds to the orientation in which the signal response is the strongest, thereby capturing the intuitive notion of orientation, a crucial ingredient for shape and general fall pattern. Furthermore, \(m_{\text{texture}}\) captures the magnitude of all orientations. Without favoring any orientation, it captures an intuitive, if rough, notion of texture where patch surface appearance and consistency matter. In Grayscale HOG, we reduce information from three color channels to effectively only one, mimicking a “grayscale” image. Of course, it is very different from the conventional notion of reducing an RGB image to a grayscale one. The name here is for analogous purpose only. Finally, in Color HOG, we genuinely retain the information of each color channel (in the form of HOG features), which is useful for us to assess how color and lack thereof affect the performance of our system.

We now convert the \(h\)-vector, where \(h = 25\) or \(75\), of each cell into a single discrete value known as a texton. Consider first a (large) set of HOG features of \(h\)-vectors collected as described above over each frame of every video in the training data set. We apply the standard \(k\)-means clustering algorithm to obtain \(k = 400\) clusters for this set of \(h\)-vectors. Each cluster corresponds to a specific texton, and we can use these 400 clusters as a dictionary to assign any \(h\)-vector to the specific texton by determining its closest cluster in the Euclidean \(\mathbb{R}^h\) space. Now, any rectangular subregion \(R\) of an image is simply a tiling of textons as shown in Supplementary Figure S4. Consider the entire region \(R_1 = R\) as shown in Supplementary Figure S5a, the 4 equally partitioned regions \(R_2, R_3, ..., R_4\) in Supplementary Figure S5b, and the 16 equally partitioned regions \(R_6, R_7, ..., R_{21}\) in Supplementary Figure S5c, all from the same image. In each of these 21 regions, we tabulate its residing textons into a 400-histogram representing the frequency of occurrence of each texton in the region. Concatenating these histograms, we see that any rectangular subregion can be represented as a \(21 \times 400 = 8400\)-vector. Finally, the histogram intersection function is applied to pairwise 8400-vectors to form a type of spatial pyramid kernel\(^1\), which can be fed directly to the RVM. This establishes our image semantics-based feature engineering.
Fig. S4 Grid of textons. Their occurrence histograms are used as image semantics-based features.

Fig. S5 Spatial pyramid of histograms. They are essentially spatially-aware bags of features.

It is important to note that numbers such as 8 (the number of discrete orientations) or 400 (dictionary size) are tunable parameters. We use concrete numbers in our discussion for only the purpose of ease of understanding and avoidance of algebraic notational clutter, without any loss of generality.

Relevance Vector Machine (RVM) Learning Framework

We now start RVM\(^2\) to train our detectors, with details of the learning framework as follows. Consider first the extracted features represented as the design matrix \(\Phi\) of size \(N \times M\), where \(N\) is the number of training instances and \(M\) is the number of features. Denote the \(j^{th}\) column by \(\phi_j\), \(1 \leq j \leq M\). We seek to find a preferably sparse \(M\)-vector \(w\) for which 

\[
y = \sigma \left( w^T \phi(x) \right),
\]

where \(\phi(x)\) is a feature vector, would approximate the probability that the data \(x\) is ham (human/a falling event) rather than spam (not human/not a falling event), with sigmoid function 

\[
\sigma(u) = \frac{1}{1 + \exp(-u)}
\]

equating the value of \(y\) to lie in the range \((0, 1)\) so that \(x\) would be classified as ham if, for example, \(y = 0.5\).

Consider a zero-mean Gaussian prior \(p(w|\alpha)\), where for each \(w_j\) of \(w\) the \(M\)-vector hyperparameter \(\alpha\) satisfies \(w_j \sim \mathcal{N}(0, \alpha_j^{-1})\) for each \(j\). Let \(t \in \{0, 1\}\) be the ham/spam ground truth indicator variable (for each training instance). As there is no known polynomial-time algorithm for maximizing the posterior 

\[
l = p(w|t, \alpha)
\]

over \(w\), we follow a standard approximation technique of using iterative reweighted least squares (IRLS) to find a local optimum using iterative updates akin to the second-order Newton method with gradient \(\nabla \ln l = \Phi^T (t - y) - Aw\) and Hessian \(\nabla \nabla \ln l = -(\Phi^T B \Phi + A)\), where \(A\) is the diagonal matrix of \(\alpha\) and \(B\) is an \(N \times N\) diagonal matrix with entries \(y_i(1 - y_i), 1 \leq i \leq N\), yielding upon proper convergence

\[
w_{\text{IRLS}} = A^{-1} \Phi^T (t - y)\] and \(\Sigma_{\text{IRLS}} = \left( \Phi^T B \Phi + A \right)^{-1}\).
Maximizing the evidence function $p(t|\alpha) = \int_w p(t|w)p(w|\alpha)dw$ by setting its derivative with respect to $\alpha_j$ to 0, we obtain the update rule $\alpha_j^{\text{new}} \leftarrow \frac{\gamma_j}{(\omega_j^{\text{IRLS}})}$, where $\gamma_j = 1 - \alpha_j^{\text{old}} \sum_{i,j}^{\text{IRLS}}$. Now, by defining $\hat{t} = \Phi w^{\text{IRLS}} + B^{-1}(t - y)$ and $C = B + \Phi \Phi^T$, the classification problem is reduced to a regression one in which the usual quality $Q$ and sparsity $S$ parameters are readily evaluated as $q_j = \phi_j^T C^{-1} \hat{t}$ and $s_j = \phi_j^T C^{-1} \phi_j$, where $1 \leq j \leq M$. Finally, we drop any feature $j$ for which $q_j^2 \leq s_j$, as it implies a divergent hyperparameter $\alpha_j$ that bounds $w_j$ around 0 with arbitrary precision. This effectively prunes the $j^{\text{th}}$ feature away and induces sparsity on $w$.

In summary, in each RVM iteration, we first run a number of subiterations for IRLS and one single update of the quality $Q$ and sparsity $S$ parameters as usual. Then, this RVM iteration is repeated for a number of times until convergence. Upon completion, $w$ tends to be sparse, which results in efficient inference on the unseen data. Note that all features are considered in each RVM iteration, so a pruned feature in one iteration may be reacquired in a later iteration. In the above learning framework, we largely omit in the discussion the index $i$ over the instances of training data to reduce notational clutter.

References

1. Lazebnik, S., Schmid, C., & Ponce, J. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. https://inc.ucsd.edu/~marni/Igert/Lazebnik_06.pdf (accessed 1/1/2020).

2. Tipping, M. E. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* 1, 211–244 (2001).
Supplementary Testing Data Figure a-w
Stumbling test on Camera One at 2 meters (Part 1)
b, Stumbling test on Camera One at 2 meters (Part 2)
c. Stumbling test on Camera One at 5 meters (Part 1)
d. Stumbling test on Camera One at 5 meters (Part 2)
e. Stumbling test on Camera One at 5 meters (Part 3)
f, Stumbling test on Camera One at 8 meters (Part 1)
g. Stumbling test on Camera One at 8 meters (Part 2)
h, Stumbling test on Camera Two at 3 meters (Part 1)
i, Stumbling test on Camera Two at 3 meters (Part 2)
Possible fall
Camera: High angle
Site: SpeedyAI Demo

Possible fall
Camera: High angle
Site: SpeedyAI Demo

Possible fall
Camera: High angle
Site: SpeedyAI Demo
### k, Stumbling test on Camera Two at 6.5 meters (Part 2)

| Possible fall! Camera: High angle Site: SpeedyAI Demo | Possible fall! Camera: High angle Site: SpeedyAI Demo | Possible fall! Camera: High angle Site: SpeedyAI Demo |
|------------------------------------------------------|------------------------------------------------------|------------------------------------------------------|
| Broadcast                                            | Broadcast                                            | Broadcast                                            |

| Possible fall! Camera: High angle Site: SpeedyAI Demo | Possible fall! Camera: High angle Site: SpeedyAI Demo |
|------------------------------------------------------|------------------------------------------------------|
| Broadcast                                            | Broadcast                                            |

---

**UCLA Medical Research**

Alarm off. Mem 4 manual off by Francy. Site: SpeedyAI Demo

Broadcast
I, Stumbling test on Camera Two at 6.5 meters (Part 3)
m, Stumbling test on Camera Two at 10 meters
n, Slipping test on Camera One at 5 meters
Fainting test on Camera One at 5 meters

Possible fall
Camera: Low angle
Site: SpeedyAI Demo

Possible fall
Camera: Low angle
Site: SpeedyAI Demo

Possible fall
Camera: Low angle
Site: SpeedyAI Demo

Alarm off. Mem 3 manual off by Francy.
Site: SpeedyAI Demo

Broadcast
p, Falling off a ladder test on Camera One at 6 meters
q, Jumping down from a desk and falling test on
Camera One at 6 meters
r, Collapsing upon standing test on Camera One at 5 meters
s, Stumbling test along the screen edge
t, Stumbling test in the presence of multiple moving people/objects on Camera 1 and Camera 2 (Part 1)
u, Stumbling test in the presence of multiple moving people/objects on Camera 1 and Camera 2 (Part 2)
Fall test behind a transparent glass window and opened mini-blinds
Other random fall tests

A 2.5 meter-high camera (Cam 2) capturing a fall at a very steep downward angle

A shoulder-height camera (Cam 1) capturing the same fall

Falling off a ladder captured by Cam 2

Jumping down from a desk and falling captured by Cam 2