SPARSE REPRESENTATIONS AND INVARIANT SEQUENCE-FEATURE EXTRACTION FOR EVENT DETECTION

Alexandru P. Condurache and Alfred Mertins
Institute for Signal Processing, University of Luebeck, Ratzeburger Allee 160, Luebeck, Germany
{*condurache, mertins*}*@isip.uni-luebeck.de*

Keywords: Event detection; Action recognition; Invariant feature extraction; Sparse classification.

Abstract: We address the problem of detecting unusual actions performed by a human in a video. Broadly speaking, we achieve our goal by matching the observed action to a set of a-priori known actions. If the observed action can not be matched to any of the known actions (representing the normal case), we conclude that an event has taken place. In this contribution we will show how sparse representations of actions can be used for event detection. Our input data are video sequences showing different actions. Special care is taken to extract features from these sequences. The features are chosen such that the sparse-representations paradigm can be applied and they exhibit a set of invariance properties needed for detecting unusual human actions. We test our methods on sequences showing different people performing various actions such as walking or running.

1 INTRODUCTION

We are interested in the detection of unusual human behavior for security and surveillance applications. Events are by definition sparse and it seems only natural to use sparse methods to detect them. The purpose of this contribution is to prove that this concept works and identify what specific requirements need to be met along the way. To detect events, we harness the discriminative nature of sparse representations. Sparse representations are tributary to the principle of parsimony. Sparsity as a data analysis and representation paradigm is currently widely investigated for various purposes like compressed sensing (Candès and Tao, 2006; Donoho, 2006), but also feature extraction (d'Aspremont et al., 2007), and data compression. Sparse representations have already been used for recognition tasks by Wright et al. (2009) and even for the recognition of human actions using data from a wearable motion-sensor network by Yang et al. (2008) or using video data (Guo et al., 2010). However, to the best of our knowledge, no link to event detection was discussed before.

In our case, we first gather during training sufficient video material to describe the actions we want to recognize. A test video is represented as a linear combination of the training data, (i.e., of the various actions in the training data) and recognized if the representation contains mostly data from one single training action. For event detection, where the training data covers the normal case, if all actions in the training set are more or less equally present in our representation we conclude we have observed an event.

To enable this procedure, for each frame a sequence-feature vector is extracted from the chunk of video consisting of the analyzed and the last $R-1$ frames, on the basis of the contours of the silhouettes of the acting person. Therefore we are able to label each frame of a test video, starting with the R'th. Working frame based gives us the possibility to refine the decision for an action, which usually extends over many frames, by considering several frame decisions. The feature-extraction process we propose here is designed from the very beginning to generate a feature space with a set of properties that are useful for event detection. These are mainly properties of invariance with respect to anthropomorphic changes, but also to Euclidian motion in the image plane. At the same time, we design our features such that they exhibit a set of mathematical properties that make them suitable for sparse-representations-based event detection.

We assume that our video data has 24 fps and use this to make several choices in our method, which may otherwise appear random. Since our main purpose is to prove that sparse representations are suited for event detection and to set the frame for further research in this direction, the data sets on which we test our algorithms is less challenging than usual, but nevertheless related to real practical scenarios (Gorelick et al., 2007).

2 SPARSE CLASSIFICATION AND EVENT DETECTION

Sparse-representation based classification bares resemblances to nearest-subspace methods, which in turn stem from nearest-neighbor classification.

Building on such premises, sparse-representation based classification looks for the sparsest representation of a test vector in terms of a dictionary of training vectors. This representation is sparse because it should contain only vectors from the class to which the test vector belongs (Wright et al., 2009).

2.1 Sparse classification

Let the training set be denoted by the matrix $\mathbf{T} = [\mathbf{T}_1, \ldots, \mathbf{T}_k]$, containing the class-submatrices $\mathbf{T}_i = [\boldsymbol{v}_{i,1}, \ldots, \boldsymbol{v}_{i,N_i}]$ with $i = 1, \ldots, k$, where N_i is the number of vectors in class i and k the number of classes. The total number of vectors in $\mathbf{T}$ is $n = \sum_{i=1}^{k} N_i$ and each vector $\boldsymbol{v}_{i,j},\ j = 1 \ldots N_i$ has m entries. Then, for each new vector $\boldsymbol{y}$ that we want to classify, we ideally have:

$$\begin{aligned} \boldsymbol{y} &= \mathbf{T}\boldsymbol{x} \\ &= x_{i,1}\boldsymbol{v}_{i,1} + \cdots + x_{i,N_i}\boldsymbol{v}_{i,N_i} \end{aligned} \tag{1}$$

where the coefficient vector $\boldsymbol{x} = [x_{1,1}, \ldots, x_{k,N_k}]^T$ of length n has entries $x_{i,j}$ different from zero only for the training-space vectors from the class i, to which $\boldsymbol{y}$ belongs. This system of equations is usually overcomplete with the number n of vectors in the training space being well larger than the dimension m of the vectors. Thus, there are infinitely many solutions to (1). Assuming equal number of training vectors per class, the more classes the more sparse $\boldsymbol{x}$. Thus we do not search for some x, but for the *sparsest* vector $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ that solves equation (1). We find it by optimizing over the ℓ^0 pseudo norm, solving:

$$\hat{\boldsymbol{x}} = \arg\min \| \boldsymbol{x} \|_0 \quad \text{subject to} \quad \mathbf{T}\boldsymbol{x} = \boldsymbol{y}. \tag{2}$$

Ideally, assuming that a vector $\boldsymbol{y}$ is represented solely with the training vectors from the correct class (that is, the components of $\boldsymbol{x}$ corresponding to training vectors from other classes are all zero), the vector $\boldsymbol{y}$ can be classified by looking up to which class the nonzero entries in $\boldsymbol{x}$ belong. In practice, of course, questions about the required amount of sparsity and the uniqueness of the sparsest solution arise. Donoho and Elad (2003) showed that if some $\boldsymbol{x}$ with less than $\frac{m}{2}$ nonzero entries verifies $\boldsymbol{y} = \mathbf{T}\boldsymbol{x}$, then this is the unique sparsest solution. This means that we have a good chance of finding the correct and unique sparsest solution to (1) even for two-class problems or for configurations in which the number of training vectors per class is not the same over all classes, provided we have enough vectors in the training set.

The decision. In practice, because the solution to equation (2) is computationally difficult to find, we solve instead:

$$\hat{\boldsymbol{x}} = \arg\min \| \boldsymbol{x} \|_1 \quad \text{subject to} \quad \mathbf{T}\boldsymbol{x} = \boldsymbol{y} \tag{3}$$

Candès and Tao (2006) showed that minimizing over the ℓ^1 norm instead of the ℓ^0 pseudo norm yields the same solution if $\boldsymbol{x}$ is sparse enough.

The vector $\boldsymbol{x}$ found after we solve (3) will usually have the largest entries for one class only, and small non-zero entries for other classes as well. Next, we use the following notation: $1_i = [b_1, \ldots, b_n]^T,\ b_l \in \{0,1\},\ l = 1, \ldots, n$ is the selection vector for class i, whose entries are everywhere zero, except for the positions of the columns of $\mathbf{T}$ that contain the training vectors of class i, where they are one and $\boldsymbol{v}_1 \odot \boldsymbol{v}_2$ is the component-wise product of two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Thus, $1_i \odot \boldsymbol{x}$ selects the entries of $\boldsymbol{x}$ where the coefficients of class i reside. To assign a class label to the vector $\boldsymbol{y}$, we use $C : \mathbb{R}^m \to \{1, \ldots, k\}$ defined as:

$$C(\boldsymbol{y}) = \arg\min_i \| \boldsymbol{y} - \mathbf{T}(1_i \odot \hat{\boldsymbol{x}}) \|_2 \tag{4}$$

such that $\boldsymbol{y}$ is assigned to the class whose training-set vectors best reproduce it (Wright et al., 2009).

2.2 Event detection

The "unsure" decision. If a test vector cannot be assigned with sufficient confidence to any of the classes represented in $\mathbf{T}$, then it receives the label "unsure". In order to express such a confidence, for the decision rule in (4), a *sparsity concentration index* is defined as

$$SCI(\boldsymbol{x}) = \frac{l \max_i \left(\frac{\| 1_i \odot \boldsymbol{x} \|_1}{\| \boldsymbol{x} \|_1} \right) - 1}{l - 1} \tag{5}$$

The vector is labeled "unsure" when $SCI(\boldsymbol{x}) \leq \tau$. The parameters l and τ need to be set empirically.

Decision for a data sequence. Until now we have discussed how to decide for a test vector. If the data we analyze is a sequence of vectors (e.g., one for each video frame) we need to adapt our decision. Then, we consider a time window of L vectors in the beginning of the sequence and decide for the class that yields a majority among these vectors, while ignoring "unsure" decisions. If all decisions are "unsure", then the sequence is classified as 'unsure".

Event detection. Should a single test vector be labeled as unsure, then this is equivalent to detecting an event, assuming the normal case is properly captured in the vectors of the training matrix $\mathbf{T}$. For our

purposes, should a sequence of vectors be labeled as unsure, then this is equivalent to detecting an unusal action, i.e., an event.

3 INVARIANT FEATURE EXTRACTION

We use here features based on Fourier descriptors (FD) (Arbter et al., 1990) computed from the contour of the acting person. Because we analyze sequences, the final sequence-feature vector, corresponding to one frame, contains information from a set of R consecutive frames, including the current one. These features are chosen such that they exhibit some invariance properties needed for event detection. A feature vector is computed for every frame of video starting with the R'th, we work thus with an overlap of $R-1$.

Contour extraction and the Fourier spectrogram. The features we extract should describe human actions. These actions take place under various illumination conditions, and are conducted by persons wearing differently-textured clothes. To achieve invariance over such conditions we extract our features from the contour of the acting person.

To find the contour, assuming the human is the only object moving in our video and the background is available, we first compute a binary motion mask B by subtracting the background from the current frame and comparing the result with a threshold (Otsu, 1979). We obtain the contour $\mathcal{C} = [(x_1,y_1),\ldots,(x_{n_C},y_{n_C})]^T$ with n_C contour points by subtracting the eroded motion mask from the original, where the erosion uses a cross-like structuring element. We compute the FDs for the contour as:$\theta(\omega_p) = \mathcal{F}(\mathcal{C}^c) = \sum_{i=1}^{n_C} u_i e^{-j\omega_p i}$ with ω is the angular frequency, $\mathcal{C}^c = [x_1 + jy_1,\ldots,x_{n_C} + jy_{n_C}]^T$ is the complex representation of the contour and $\mathcal{F}$ is the Fourier operator. As features ϑ^f for frame f we keep only the magnitudes of Q descriptors – half for the negative and half for the positive frequencies, excluding the DC component: $\vartheta^f = [|\theta_1|,\ldots,|\theta_Q|]^T$

We want to extract sequence-features, for which purpose we introduce next the Fourier descriptors spectrogram (FDS). Considering the data on which we demonstrate our algorithm, we assume the analyzed actions have a certain periodicity, extending over at most R frames. Thus, different time windows of length R from the video of the same person, conducting the same action, contain chunks of the same periodic signal, but with various phases. To compute our sequence-feature vector, we gather the frame-feature vectors from R frames in a matrix with Q lines and R columns. We call this matrix the Fourier descriptors spectrogram: $FDS = \left[\vartheta^1,\ldots,\vartheta^R\right]$. Using only the magnitudes of the FDs, the FDS is invariant to several operations, including starting point, rotation and translation.

3.1 Invariant sequence features

Viewpoint changes, scale variations but foremost the anthropometric characteristics of various persons lead to changes in the acquired contours. Our method should be invariant to such changes. To introduce the needed invariances into our algorithm, we make use of invariant integration as described by Schulz-Mirbach (1992). Invariant integration returns features invariant to the actions of a group of transformations on an input signal. For this purpose we define a feature function $f(\cdot)$ on the signal space and integrate it over the group of actions. Schulz-Mirbach (1994) shows that the set of monomials $m(\cdot)$ is a good choice for $f(\cdot)$. For a D-dimensional input $\mathbf{t} = [t_1,\ldots,t_D]$, the monomials are defined as $m(\mathbf{t}) = \prod_{d=1}^{D} t_d^{b_d}$, where $b_d \in \mathbb{N}$. Invariant integration is a powerful tool with a wide range of applications including the analysis of observation sequences, as shown by Müller and Mertins (2011) for speech data.

A model for anthropometric changes. We define the group of transformations to which we achieve invariance in relation to the effects that anthropometric changes have on the contour of the person. At this stage, we model anthropometric changes by the convolution of the contour with pairs of Dirac pulses, where the distance between the two pulses is variable. Thus, we would like to achieve invariance to sinusoids modulating the FDs. Multiplication of the FDs with a sinusoid is equivalent to a shift of the Fourier coefficients of the FDs. Therefore, we need to compute features invariant to shifts of the Fourier transform of the FDs. We first compute the Fourier transform of the columns of FDS, obtaining thus: $\mathcal{S} = \mathcal{F}_c(FDS) = [\varphi^1,\ldots,\varphi^R]$, with $\varphi = [\phi_1,\ldots,\phi_Q]^T$. Next, we apply invariant integration on the columns of $\mathcal{S}$ and consider only the magnitudes. By integrating over all shifts from 1 to Q, while enforcing suitable boundary conditions, we achieve invariance to a set of modulating sinusoids of various frequencies.

The feature function. For our purposes, we use monomials of order two. Thus, $b_d = 0$, for $d \in \{1,2,\ldots,Q\}\backslash\{q_1,q_2\}$ and $b_d \neq 0$ for $d \in \{q_1,q_2\}$. For better separability, various monomial features

(i.e., different values for q_1 and q_2) are used, obtaining for each column of $\mathcal{S}$ an invariant feature vector with one entry per monomial. Here, we need a scalar feature for each column from $\mathcal{S}$, thus, we define our feature function $\mathcal{M}(\cdot)$ to be a linear combination of monomials. To obtain class-conditional distributions and thus a feature space that is better suited for the sparse classifier, we chose $b_d \in \mathbb{R}$, instead of $b_d \in \mathbb{N}$.

Our feature function consists of a linear combination of three monomials m_i, $i = 1 \ldots 3$, from various entries along the columns φ^r, $r = 1, \ldots, R$ of $\mathcal{S}$. For m_1, we use $q_1 = q$ and $q_2 = q-1$ with $b_{q_1} = b_{q_2} = 1.7$, for m_2 we use $q_1 = q$ and $q_2 = q-3$ with $b_{q_1} = b_{q_2} = 1.5$ and for m_3 we use $q_1 = q$ and $q_2 = q-5$ with $b_{q_1} = b_{q_2} = 1.3$. The index q runs over the entries of φ^r. The feature function is then: $\mathcal{M}(\varphi^r; q) = (\phi^r_q \phi^r_{q-1})^{1.7} + (\phi^r_q \phi^r_{q-3})^{1.5} + (\phi^r_q \phi^r_{q-5})^{1.3}$, $q = 1, \ldots, Q$, with periodic boundary conditions.

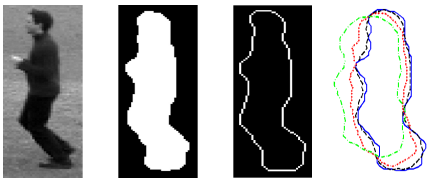
To compute the sequence feature we should integrate over this feature function. Integration is numerically approximated by summation, which is in turn unnormalized mean computation. For our features we use more robust order statistics instead of integration. The sequence-feature vector is computed by taking the 25'th percentile over the combinations of monomials: $v_r = p_{25}(\mathcal{M}(\varphi^r; q))$. Therefore, for the entire $\mathcal{S}$, we get, an R-dimensional sequence-feature vector $\boldsymbol{v} = [v_1, \ldots, v_R]^T$ that is invariant to a set of anthropometric variations.

The parameters of the feature function and the precise percentile were empirically chosen on the dataset we've used to test our methods.

4 EXPERIMENTS

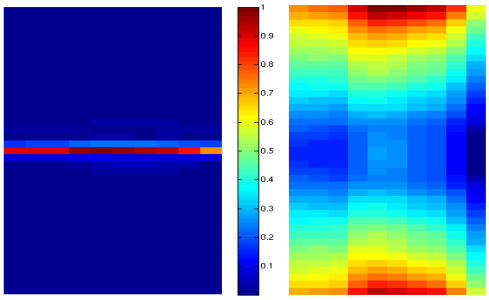
We test our methods on part of the KTH action database (Schuldt et al., 2004). This database contains six types of human actions (walk, jogg, run, box, hand wave and hand clapp) performed by 25 persons. We use the walk (W), jogg (J) and run (R) actions. Each person performs the action four times: three times outdoors and one time indoors. We have used the outdoor sequences where the person moves parallel to the camera. Therefore, for each action we have 25 action-sequences (one for each person) in our data set. We extract one sequence-feature vector per frame of video obtaining a classification result for each analyzed frame. The decision for a multi-frame action-sequence is taken as for a data sequence.

We divide our experiments into: feature extraction, action-sequence recognition and point-event detection. For action-sequence recognition, the query action sequence is already in the training set – this would also be the scenario in which these methods are used as a building block for context and collective event detection algorithms. For point-event detection, the analyzed action sequence is not in the training set and is significantly different from other action sequences in the training set. To compute the *SCI*, we use equation (5) with $l = 10$. For the "unsure" decision we use $\tau = 0.4$.



(a) (b) (c) (d)

Figure 1: Motion region in a frame (a), motion mask (b) and its contour (c). Original contour (continuous blue line) and and variations to which we are invariant after invariant integration (interrupted lines), applied to the $\mathcal{S}$ (d)



(a) (b)

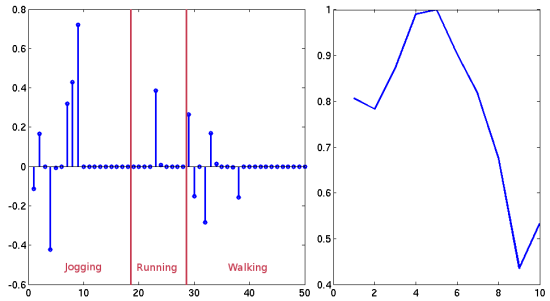
Figure 2: *FDS* (a) and $|\mathcal{S}|$ (b)

4.1 Feature extraction

After motion detection, we obtain a motion mask (Figure 1 (b)). That we use to detect the contour of the moving person (Figure 1 (c)). The contour is further used to compute the FDs with $Q = 40$ and the FDS (Figure 2 (a)) with R=10 that in turn yields $|\mathcal{S}|$ (Figure 2 (b)), and after the invariance transform, our feature vector (Figure 3 (b)). By using the magnitudes of the FDs we are already invariant to a set of variations of the original contour. The invariance transform achieves that the corresponding feature vector is invariant to several more contour variations. Some of these additional variations are shown in Figure 1 (d).

4.2 Action recognition

We have conducted two experiments. In the first experiment we look at how is each video frame classified. In the second we use the frame decisions to clas-



(a) (b)

Figure 3: The coefficient vector for a frame from a jogging sequence with decision regions (a), our sequence-feature vector (b).

(a) Results including "unsure" frames

(%)	J	R	W	unsure
J	36.79	3.28	18.99	40.94
R	0	75	0	25
W	2.72	0.98	65.38	30.92

(b) Results ignoring "unsure" frames

(%)	J	R	W
J	62.36	5.56	32.19
R	0	100	0
W	3.84	1.42	94.75

Table 1: Permutation matrices for frame decisions.

sify action sequences, with $L = 24$. The results were computed by means of the leave-one-out method. The procedure was repeated until every point from the available data set was used for testing once. To compute the training matrix, we may use for example, for jogging the first nine, for running the first five and for walking the first eleven frames of each of a certain number of sequences per action type. A coefficient vector resulting from a training space with two action sequences per action type is shown in Figure 3 (a).

We have computed the permutation matrices for the types of sequences we have worked with. The permutation matrices should be read along lines, e.g., for the action of jogging, the first column contains correct decisions, the second wrong decisions in favor of the class labeled "Running" and the third column wrong decisions in favor of the class labeled "Walking". For the frame experiment, on average 32% of all frames in a sequence are labeled as "unsure". In Table 1(a) we show results including the "unsure" frames and in Table 1(b) we show results ignoring the "unsure" frames. For the sequence experiment, the results are shown in Table 2. For the sequences labeled as "unsure", all first 24 frames were labeled as 'unsure'.

(%)	J	R	W	unsure
J	86.67	0	13.33	0
R	0	60	0	40
W	0	0	100	0

Table 2: Permutation matrix for action-sequence decisions.

Normal	J & R	J & W	R & W
Event	W	R	J
(%)	100	73.33	86.67

Table 3: Event-detection results for the first experiment.

4.3 Event detection

For event detection we have conducted two experiments: for the first one, we have used two types of actions to simulate the normal case and the event was the third (e.g., running and walking were normal and jogging was an event) and for the second one, we have used one type of action as normal case, and the other two were the event. The results for the first experiment are shown in Table 3 and for the second, in Table 4. These tables show the percentage of sequences correctly classified as event. The results are obtained by a modified type of five fold cross-validation. At each iteration, the training matrix is composed of five different "normal" action-sequences and the event set is given by all 25 "event" action-sequences. In the end we compute the average of the detection rates.

5 DISCUSSION CONCLUSIONS AND SUMMARY

Since a step during walking takes some ten frames, we choose the number of columns of the FDS to be $R = 10$. For each of the $L = 24$ frames needed to take a sequence decision, a feature vector is extracted from ten frames (nine previous frames and the current one). Thus, a decision for a sequence is taken after 34 frames have been recorded. Clearly, to analyze a sequence in the current setup, a minimum of at least ten frames is needed, in which case the decision for the 11'th frame is the decision for the entire sequence. Extracting the sequence features from ten frames is well suited for the current data. The number of frames to be considered for a sequence-feature vector should be chosen in relation with the frame rate of

Normal	J	R	W
Event	R & W	W & J	R & J
(%)	96.67	100	86.67

Table 4: Event-detection results for the second experiment.

the analyzed video and the length of an atomic part of the analyzed action (in this case one step of the person executing the action) and validated on the training data. Deciding for a sequence based on a majority of frame-decisions from the first 24 frames proved well suited for the current data set. As a rule of the thumb, the more frames are considered, the better the sequence-decision. During feature extraction, we use $Q = 40$ FDs left and right from zero and hence implicitly assume a minimal contour length for a certain image resolution. This number of FDs is well suited for our data, but it should be chosen according to these considerations in practice. All parameters with no rules for determining them were established by six-fold cross validation using the videos of the first 10 persons. Our feature vector is invariant to several variations of the person's contour, some corresponding to anthropometric changes, however, some can be thought of as corresponding to viewpoint changes and to scale changes and thus we obtain also a mild viewpoint invariance in our feature vector.

A decision for a sequence of around 50 frames is available after 47 seconds under MATLAB on a 2.66 GHz dual-core machine. However, many of the algorithmic steps can be conducted in a parallel manner.

The sparse classification paradigm can be used for action recognition and to detect all sort of point events. While not directly suited for context and collective events, it may represent an action-recognition building block for such algorithms, other blocks being necessary for analyzing the chains of individual actions (Matern et al., 2011). The particular feature extraction process we use here is specific for the analysis of human behavior. We are concerned with the behavior of a person in a single track and the current feature extraction is adapted for this case. A prerequisite for deploying these methods in more complicated scenarios is a successful tracking, irrespective of the number of cameras used.

Our algorithm can be seen of consisting of two parts: feature extraction and sparse classification. The feature extraction is targeted to certain invariances and tailored to the sparse classifier. We have shown that sparse classification as introduced by Wright et al. (2009) is well suited for human event detection. Sparse classification offers a set of advantages over other methods for the problem of action recognition and event detection, being robust, adaptive and easy to tune. In this context, the focus is now set on the extraction of suitable features to enable the usage of such methods. Furthermore, even if the issue of invariance can be addressed at the classifier level when using sparse classifiers, many of the desirable invariance properties that characterize a good human action recognition/event detection method should be obtained by means of the feature extraction process. We have shown how to use the invariant integration as described by Schulz-Mirbach (1992) to extract such features from the contour of the acting person.

REFERENCES

Arbter, K., Snyder, W., Burkhardt, H., and Hirzinger, G. (1990). Application of affine-invariant fourier descriptors to recognition of 3-d objects. *IEEE Trans. Patt. Anal. Mach. Intell.*, 12:640–647.

Candès, E. and Tao, T. (2006). Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inform. Theory*, 52(12):5406–5425.

d'Aspremont, A., Ghaoui, L. E., Jordan, M., and Lanckriet, G. (2007). A direct formulation of sparse pca using semidefinite programming. *SIAM Review*, 49(3).

Donoho, D. (2006). Compressed sensing. *IEEE Trans. Inform. Theory*, 52(4):1289–1306.

Donoho, D. and Elad, M. (2003). Optimal sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization. *Proc. Nat'l Academy of Sciences*, pages 2197–2202.

Gorelick, L., Blank, M., Shechtman, E., Irani, M., and Basri, R. (2007). Actions as space-time shapes. *Trans. Patt. Anal. Mach. Intell.*, 29(12):2247–2253.

Guo, K., Ishwar, P., and Konrad, J. (2010). Action recognition using sparse representation on covariance manifolds of optical flow. In *Proc. AVSS*, pages 188–195.

Matern, D., Condurache, A. P., and Mertins, A. (2011). Event detection using log-linear models for coronary contrast agent injections. In *Proc. ICPRAM*.

Müller, F. and Mertins, A. (2011). Contextual invariant-integration features for improved speaker-independent speech recognition. *Speech Comm.*, 53(6):830 – 841.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Trans. on Sys., Man and Cyb.*, SMC-9(1):62–66.

Schuldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: A local svm approach. *Proc. ICPR*, 3:32–36.

Schulz-Mirbach, H. (1992). On the existence of complete invariant feature spaces in pattern recognition. In *Proc. ICPR*, volume 2, pages 178–182, The Hague.

Schulz-Mirbach, H. (1994). Algorithms for the construction of invariant features. In *DAGM Symposium*, volume 5, pages 324–332, Wien.

Wright, J., Yang, A., Ganesh, A., Sastry, S., and Ma, Y. (2009). Robust face recognition via sparse representation. *IEEE Trans. Patt. Anal. Mach. Intell.*, 31(2):210–227.

Yang, A., Jafari, R., Sastry, S., and Bajcsy, R. (2008). Distributed recognition of human actions using wearable motion sensor networks. *J Amb. Intl. Smt. Env.*, 30(5):893–908.