Product Manifold Regression for Action Recognition and Beyond

Final Implementation, 3 Team Members

1 Abstract

Action videos are multi-dimensional data that can be represented via data tensors. The goal of this project is to characterize data tensors as points on a product manifold and perform regression for action recognition. To fulfill this goal, we first decompose the tensor data using Higher Order Singular Value Decomposition (HOSVD), and impose the unfolded matrices on a Grassmann manifold. We then formulate a regression function to model the processed data for learning and classification without explicitly modeling the shape and motion. Databases used in related papers are employed, namely the Weizmann Actions as Space-time Dataset and UC San Diego Traffic Video database, to determine the efficacy of this approach and to challenge the implementation as originally described in the literature. Our experimental results show the efficiency and accuracy of the method we implemented; a second method is implemented based on earlier literature to demonstrate the strength of manifold regression as compared to simpler manifold methodologies.

2 Motivation

Human gestures and actions are a vital part of daily human lives; while humans instinctively recognize such movements, quick and automated gesture and action recognition can be very difficult, in no small part due to the multidimensional nature of video data. Effective methods to identify human gestures can drastically expand the way people interact with computers in their day-to-day lives, and the video-reliant nature of this task is widely-studied. As discussed in Lui [1], one common problem of several gesture recognition methods, as we will discuss in detail in later sections, is the need for large amounts of training data or time to have a result with good accuracy. The method we found proposes a solution to optimize the process of characterization and classification over this multidimensional data via regression; in contrast to complex models such as deep learning methods, regression potentially allows for faster training and classification, marking these less-complex models as worthy of further study. In addition, to study how this method can be more generally applied to the problem of action and video recognition and classification, we implemented and compare both methods upon the UC San Diego Traffic Video database ([2]).

3 Problem Statement

The goal of our project is to implement a method in the paper by Lui [1] which proposes an approach to human gesture recognition based on the use of tensors and the geometry of product manifolds. An important note made by the authors of the original paper is the applicability of their proposal upon other visual applications that we can test with our implementation. The main novelty that comes with this implementation is the ability to generalize their implementation to day to day use cases. The method does not rely on extensive training or parameter tuning, doesn't make any explicit assumptions on the data, and still can detect and use appearance and motion without needing to model shapes explicitly. All this leads to a simple pixel-based representation of action data while maintaining equal, if not better, performance to other gesture recognition methods. All the novelties of this implementation go back to idea of the importance of quick and automated gesture recognition to real world applications, which this method is tuned towards.

4 Related Works

The author of the original paper highlights a variety of techniques for action/gesture recognition in recent years, including bag-of-features models, autoregressive models, 3D Fourier transforms, tensor frameworks, and product spaces. Bag-of-features models ([3–5]) wherein visual vocabularies are learned from feature descriptors and spatiotemporal features that are usually represented by a normalized histogram provide encouraging results yet have heavy training load for classification and may consume a shockingly large amount of time. Another avenue of action classification is to use an autoregressive (AR) model. This approach has been used for dynamic texture recognition ([6]), human gait recognition ([7]), modeling human silhouettes ([8]) based on Kendall's theory of shape ([9]) and human activity analysis ([10]). Shapes are expressed on a shape manifold and the dynamics of a human image is modeled. The three-dimensional Fourier transform has been demonstrated by Weinland et al. [11] and Rodriguez et al. [12] as a valuable tool for action classification, although action matching of the method is view invariant so training videos need to be acquired from multiple cameras. Product space can used to train a motion manifold in Datta et al. [13], and a likelihood framework is employed in Lin et al. [14] and find optimal action alignment in Li and Chellappa [15]. Data tensors are the multidimensional generalization to matrices that is close to the method implemented in this paper: the appearance is captured by the basis images and the dynamics are encoded with a weighting vector. Besides product spaces, which have not been directly considered in the previously-mentioned works, the highlighted techniques are directly used for human action recognition and training.
Besides the work that Lui [1] refers to in his paper that we choose to implement, there are also papers by Lui and his team regarding his algorithm on manifolds and human gesture recognition. One paper we found highly useful to the project is Lui et al. [16] where he provides mathematical background and interpretations of some concepts mentioned in Lui [1]. Another paper we found helpful is a literature review done on the implemented paper, Michalczuk et al. [17]. It provides some deep insights of the paper Lui [1] and gives a general description of how to calculate and interpret the Karcher mean instead of the pseudo-code in Lui, though there are some conflicts between the two papers. The final implementation of our manifold calculations are based on the closed-form formulation presented in Wang et al. [18].

5 Method

The method behind this implementation turns action videos (the data) into tensors and finds the relations between the tensors using a product manifold (in this case used to describe the product of our factor manifolds). The implementation relies on the fact that the geodesic (the shortest path between two points in a surface) of a product manifold is the Cartesian product of the geodesics of each factor manifold that will be created. The action videos are decomposed into three factor manifolds using Higher Order Singular Value Decomposition (HOSVD), the product of which comprises the product manifold. In order to fit the elements from a training set the implementation uses a form of least squares regression on the factor manifolds. However, since normal least squares regression works on Euclidean space, the implementation uses a composite function that ensures values generated by the least squares regression reside on a manifold rather than in Euclidean space.

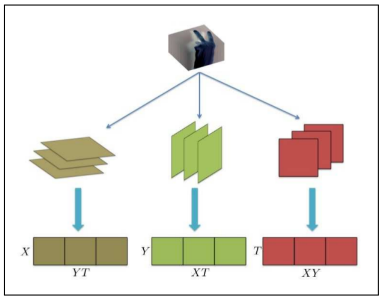


Figure 1: Matrix Unfolding an action video with spatial dimensions *X*, *Y* and temporal dimension *T*.

5.1 Obtaining Factor Manifolds

The first step in obtaining Factor Manifolds is to perform Matrix Unfolding upon our video sequence. Matrix unfolding takes a tensor of order *N* and maps it to a set of *N* matrices such that the *i*th matrix in that set is a lower-mode matrix of the tensor. In Figure 1, obtained from Lui [1], we can see an example of this performed upon an action video, where the third-order tensor is sliced along each of its dimensions, *X, Y, T*. The slices of each dimension are then concatenated together such that the rows represent a single change in that dimension and the columns represent the subsequent change in the other two dimensions.

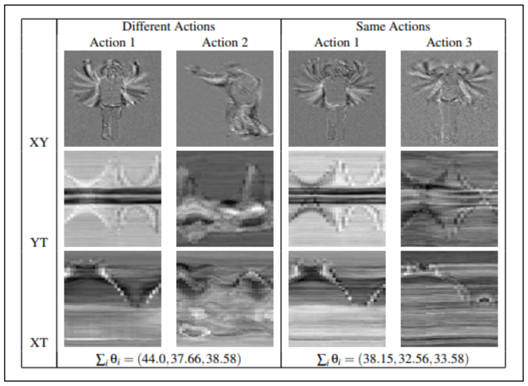
After obtaining the *N* unfolded matrices from an *N*-order tensor, HOSVD is performed on the tensor. HOSVD on an *N* order tensor is done as

$$A = S \times_1 V^1 \times_2 V^2 ... \times_n V^n$$

where *S* is a core tensor of singular values and V^n is the orthogonal matrix spanning the column space of the *n* unfolded matrix, and $\times_k$ is mode-*k* multiplication. This leaves us with *n* Factor Manifolds for a tensor of order *n*. Note that although the core tensor is difficult to calculate in general, we do not need to do so for the sake of this implementation, and thus can bypass a great deal of computation time by not doing so.

5.2 How Product Manifolds Work With Tensors

A product manifold can simply be stated as a composition of multiple lower dimensional manifolds. In this implementation's case, the product manifold is a composition of the Factor Manifolds. This product manifold exhibits the variations both spatially and temporally in the action video as the factor manifold that changes over time shows the resulting change in appearance, the second factor manifold over *X* shows vertical motion, and the third factor manifold over *Y* shows horizontal motion. This means that the distance on the product manifold is the distance in terms of appearance and motion.



Different Actions
Action 1
Action 2
Same Actions
Action 1
Action 3
XY
YT
XT
$\sum_i \theta_i = (44.0, 37.66, 38.58)$
$\sum_i \theta_i = (38.15, 32.56, 33.58)$

Figure 2: Depiction of motion and appearance changes from a set action compared to a different action and to the same action.

Figure 2 from Lui [1] depicts a visualization of the product manifold. The first column shows the depiction of two different actions in terms of the factor manifolds, the first being in terms of *T*, the second being in terms of *X*, and the third being in terms of *Y*. Column 2 depicts the same but where the actions are the same. It is clear that the factor manifolds of the same action look much more similar than those of differing actions. This is also quantified at the bottom of the figure with the sum of canonical angles for each factor manifold. The distance between Grassmann manifolds is characterized on a product manifold using canonical angles θ. The canonical angle is computed by the factorization of the canonical variate, which are just the images we can see for each action. Simply put, the smaller the canonical angle the more similar the two actions are, and Action 3 has a lower sum canonical angle compared to Action 2.

5.3 Applying Least Squares Regression

Linear Least Squares Regression only works in Euclidean space, so we need to transform the regression problem to fit a manifold. We first change the least squares regression formula to a non-linear form, which is done through a kernel function.

$$A(A * A)^{-1}(A * y)$$

The * is an RBF kernel described in Lui [1], where

$$x * y = exp(-\frac{\sum_k \theta_k}{\sigma})$$

where x, y are the row and column elements of a factor manifold, θ_k is the canonical angle mentioned earlier and, for our method, σ is 2. As we have three factor manifolds, this regression problem turns into three sub-regression problems, one for each type of factor manifold. These sub-regression estimators give a weighting vector for the training data on the specific factor manifold. However, these estimators give the weighting vector that exists in a vector space; in order to connect the weighting vector in vector space with the factorized training data in the factor manifold we need to map points from the vector space to the manifold. To do this the implementation uses Weighted Karcher Mean Computation which is shown in the algorithm below (from Lui [1]).

Algorithm 1 Weighted Karcher Mean Computation

```
1: Initialize a base point $\mu$ on a manifold
2: while not converged do
3:     Apply the logarithmic map to the training samples $Y_i$ to the base point $\mu$
4:     Compute the weighted average on the tangent space at the base point $\mu$
5:     Update the base point $\mu$ by applying the exponential map on the weighted average
6: end while
```

The canonical angles can be calculated recursively by:

$$\theta_k = \cos^{-1}(x_k^\top y_k)$$

$$x_0 = V_A, y_0 = V_B$$

Gesture recognition is achieved by choosing the classification that minimizes the chordal distance between A and B, defined as:

$$dist_c(A, B) = \| \sin(\Theta) \|^2$$

Where Θ is a vector of the canonical angles between the three factor manifolds of A and B. This process (outlined in Figure 3, from Michalczuk et al. [17]) is described in Lui et al. [16], which describes a nearest-neighbor classifier based on classifying A as the same class as whichever B minimizes this chordal distance; the regression method in Lui [1] improves upon this method using least squares regression, but we have also implemented this 1NN method for comparison with our implementation.

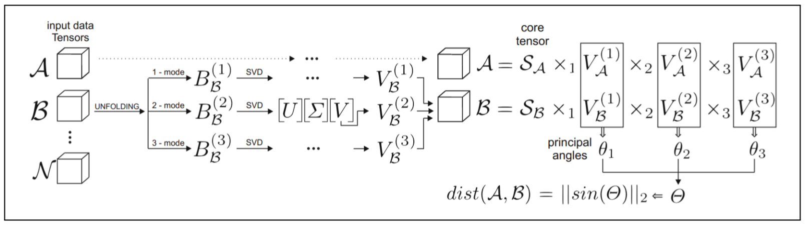

input data Tensors
$\mathcal{A}$
$\mathcal{B}$
$\mathcal{N}$
UNFOLDING
1 - mode
2 - mode
3 - mode
$B_{\mathcal{B}}^{(1)}$
$B_{\mathcal{B}}^{(2)}$
$B_{\mathcal{B}}^{(3)}$
SVD
$[U][\Sigma][V]$
$V_{\mathcal{B}}^{(1)}$
$V_{\mathcal{B}}^{(2)}$
$V_{\mathcal{B}}^{(3)}$
core tensor
$\mathcal{A} = \mathcal{S}_{\mathcal{A}} \times_1 V_{\mathcal{A}}^{(1)} \times_2 V_{\mathcal{A}}^{(2)} \times_3 V_{\mathcal{A}}^{(3)}$
$\mathcal{B} = \mathcal{S}_{\mathcal{B}} \times_1 V_{\mathcal{B}}^{(1)} \times_2 V_{\mathcal{B}}^{(2)} \times_3 V_{\mathcal{B}}^{(3)}$
principal angles
θ_1
θ_2
θ_3
$dist(\mathcal{A}, \mathcal{B}) = ||sin(\Theta)||_2 \Leftarrow \Theta$

Figure 3: Action classification on product manifold schema *X*, *Y* and temporal dimension *T*.

In Lui [1], the nonlinear operator (*) described above is the composition of an logorithmic and exponential mapping, allowing us to map from the Grassmann factor manifolds to a vector space, perform the regression, and then map back onto the Grassmann manifold; however, this requires the data to be modified to be reflectionally and rotationally invariant. Due to a lack of documentation in literature regarding the implementation of this iterative formulation and our unfamiliarity with differential manifold topology, we instead implemented an closed-form solution to this problem described in Wang et al. [18], which avoids the need for an iterative algorithm by embedding the elements of the Grassmann manifolds to the space of symmetric matrices, which is equivalent to obtaining the Karcher mean; this approach has the added advantage of avoiding potential losses when approximating the Karcher mean using the iterative process in Lui [1], at the expense of additional calculation time. Rather than projecting data from a Grassmann manifold onto a vector space, performing regression, and then projecting them back onto the Grassmann manifold, we instead use a simple map to project X on a Grassmann manifold onto the space of symmetric matrices:

$$M(X) = XX^{\top}$$

We can then use closed-form expressions for distance based on the Frobenius norm in the space of symmetric matrices, rather than the product manifold chordal distance, for regression. The process as described in Wang et al. [18] is as follows: let $D^k = \{D_1^k, \dots, D_n^k\}$ be the training set for a single class k, where each D_j^k is a Grassmann factor manifold; let X be the video being classified, and let X_m be a mode-m Grassmann factor manifold of the video being classified; and let $\|\cdot\|_F$ be the Frobenius norm. Then, we define matrix:

$$[K(D^k)]_{i,j} = \|D_i^{k\top} D_j^k\|_F^2$$

and the vector:

$$[K(X_m, D^k)]_j = \|X_m^{\top} D_j^k\|_F^2$$

So as to calculate the optimal regression coefficients:

$$y^* = 2(K(D^k) + K(D^k)^{\top})^{-1} K(X, D^k)$$

where we have made use of the properties of the symmetric embedding of $K(D^k)$ to ensure that the inverse exists. These regression coefficients y^* minimize the loss function over the kth class for a single mode m:

$$L(D^k, X_m) = \|X_m X_m^{\top} - \sum_{j=1}^{n} D_j^k D_j^k y_j^* \|_F$$

We can then calculate the total loss for X in the kth class on the training sets D^k, E^k, and F^k (each a set of a single mode of the unfolded and decomposed training tensors) by taking the sum:

$$\epsilon^k = L(D^k, X_1) + L(E^k, X_2) + L(F^k, X_3)$$

Finally, we see that the classification of X is predicted to be the class k^*, where:

$$k^* = \arg\min_k \epsilon^k$$

or, in other words, the class which minimizes the total loss. Note that this method bypasses the need to calculate the chordal distances, as the classification is performed within the space of symmetric matrices and not on the Grassmann product manifold itself Wang et al. [18].

6 Evaluation

6.1 Evaluation Goals

Most comparative models to the one we implemented use the same test datasets as each other, meaning for the evaluation on our novel datasets don't have similar comparative models to compare results against. In order to have data to compare against our evaluation plan is to test two models: the first being the 1NN model that is the basis of the Least Squares model, which is the other model we will evaluate. The goal of doing this is to show the effect in terms of performance that the novel contribution of this project, the Least Squares Regression, has upon classification as compared to an instance without it (1NN). By doing this, we can show any improvements found between the two models are a direct result of the novel contribution of this paper. In terms of performance metrics, we followed the same performance metrics they used in the paper which is classification accuracy. In order to thoroughly test the implementation provided in the paper we chose two datasets that would challenge and push the limits of what this implementation can accomplish.

7 Evaluation

7.1 Datasets

A great deal of public human activity and gesture video data is presented and discussed in Chaquet et al. [19].The first dataset is the Weizmann Space - Time dataset [20]. The data consists of ninety videos, comprising ten simple actions(walk, run, jumpy, gallop sideways, bend, one-hand wave, two-hand wave, jump in place, jumping jack and skip) performed by nine different people as shown in Figure . The videos are filmed at 180x144 from a far distance from the camera. The goals of using this dataset were to see two things: 1.) How the model performs with limited training samples and 2.) How the model performs on a dataset where the geometric differences between actions are relatively vague. The second dataset that we chose is the Traffic Video Database provided the Statistical Visual Computing Lab at UCSD [2]. The traffic video database consists of 254 video sequences of highway traffic in Seattle, collected from a single stationary traffic camera over two days. The database contains a variety of traffic patterns and weather conditions (e.g. overcast, raining, sunny, rain drops on the camera lens) as shown in Figure . Each video was recorded in color with a resolution of 320×240 pixels with between 42 to 52 frames at 10 fps. A point made my Lui [1] is how the proposed implementation is not limited to just recognition tasks on human gestures and actions, but can also be used for a variety of computer vision tasks. Our goal was to put that to the test by using this dataset. The label distribution of the dataset is not equal with light having 165 instances versus heavy and medium having 44 and 45 respectively. In this evaluation we wanted to see how the inter-class accuracy is affected by the uneven label distribution on actions that the model was not originally designed exactly for.

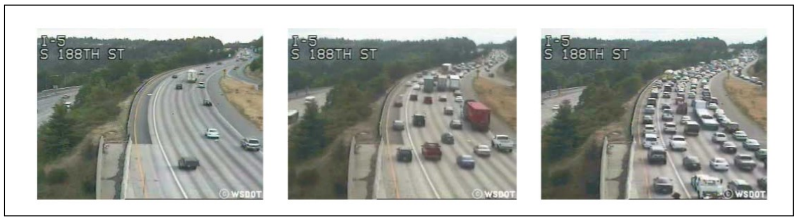


Figure 4: Example of Data from UCSD Traffic Database

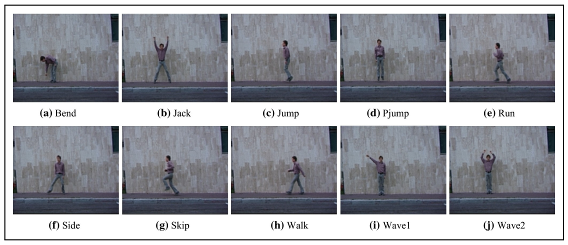


Figure 5: Example of Data from Weizmann Space - Time Dataset

7.1.1 Dataset Testing Procedures

All videos used in the implementation were resized to a height of 50 x 50 x 35 (width x height x number of frames) to standardize the amount of data given to each model between datasets. The 35 frames chosen were extracted from the middle of each video, and any video shorter than 35 frames would loop until the 35th frame. We performed no extra pixel intensity normalization or mean image subtraction or use of a sliding window as our goal was see how the pure model would perform on the datasets without other factors. For testing the Weizmann dataset we used Leave One Out (LOO) Cross Validation in which for every iteration the model is trained on the actions of all but one person and is then used to test on that person. This is a common testing procedure used in the original paper and many other human gesture/action recognition papers. For the UCSD Traffic database we use a different testing procedure since here are no human subjects and one of our goals is to see performance on different label distribution and there are no human subjects. We measure the accuracy of the model across 5 splits in which the dataset is split into 70% training and 30% testing randomly.

7.2 Results

7.2.1 Weizmann Dataset

Method:	Accuracy
1NN	67.77%
Prod Man Reg	75.55%

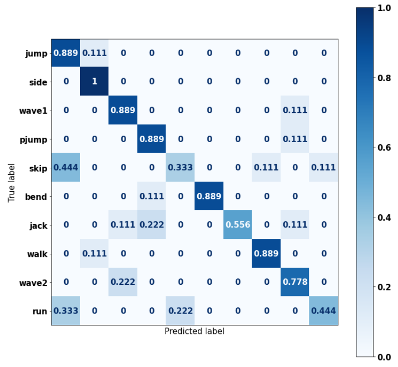


Figure 6: Confusion Matrix of the Product Manifold Regressor

The Product Manifold Regression Model performed on average around 8% better than the 1NN model. This shows that weighting classes to the best representative point for that class is especially helpful when the clusters of actions could be relatively close together due to the similar geometry. When we examined how the video data gets represented on our factor manifolds for this dataset we found that the videos naturally did not represent well on the manifold especially on the XY (appearance) manifold. This is the reason behind the relatively lower accuracy for this dataset. As we hypothesized certain actions in this dataset that are geometrically similar in their movement and dynamics, achieve poorer classification. From the confusion matrix, we can see that the actions that achieved the worse classification were run and skip, which both had high correlation to jump (person jumping across the screen) since geometrically they are very similar. To quantify this we took the average geodesic distance ($\|\sin(\theta_1, \theta_2, \theta_3)\|$) between jump and skip points and jump and run points and got 1.34 and 1.51 respectively. To give context to those numbers the average geodesic we found between a single run instance and the other run instances was 1.32. This means that the factor manifolds for those actions were very close to each other making classifications between points hard even with the optimal representation of them with the Product Manifold Regressor.

7.2.2 UCSD Traffic Database

Run:	Light Traffic	Medium Traffic	Heavy Traffic	Total
Run 1	95.5%	20%	68.8%	79.4%
Run 2	100%	35.7%	76.7%	85.3%
Run 3	98.6%	22.2%	71.4%	81.4%
Run 4	100%	21.4%	62.5%	83.3%
Run 5	95.5%	21.0%	78.6%	82.3%
Average	**97.9%**	**24.0%**	**71.6%**	**82.3%**

1NN Performance on Traffic Database

Run:	Light Traffic	Medium Traffic	Heavy Traffic	Total
Run 1	100%	45.5%	86.6%	89.5%
Run 2	98.2%	66.6%	100.0%	94.7%
Run 3	98.0%	75.0%	92.1 %	92.1%
Run 4	97.8%	60.0%	88.1%	88.1%
Run 5	98.2%	60%	92.1%	92.1%
Average	**98.4%**	**61.4%**	**91.78%**	**91.3%**

Product Manifold Regression Performance on Traffic Database

On average the product manifold regressor performs around 9% better than the 1NN regressor, but taking a more in depth it performs much more robustly. As we predicted light traffic had by far the highest percent accuracy due to the higher percent distribution of its label. The 1NN classifier is highly susceptible to this issue since there just exists a higher number of points of light traffic, meaning that if the clusters for the classes isn't clearly separable then the model is prone to choosing the classes with more instances. With the Product Manifold Regressor, the model chooses an optimal point to represent each class so this is not any issue, but instead with more data that point becomes more representative of what light traffic would look like. In terms of medium and heavy traffic, the 1NN classifier struggles much more, especially with medium. The Product Manifold Regression is much more robust for those cases, which is why its 91.3% accuracy is more indicative of its actual performance than the 1NN model whose accuracy was boosted heavily due to the classification on light traffic. We found that there were no instances of light traffic being classified as heavy and vice versa which is telling on why the performance on medium traffic is worse. Since medium traffic lies in the middle of the two other groups, this model struggles to classify in between two clear cut groups.

8 Conclusion

In conclusion, we successfully extracted two different implementations from the original implementation in Lui [1], allowing us to demonstrate the advantages of the regression classifier over a simpler method. Our project highlights well the goal of the paper in terms of improving the accuracy via least-squares regression. On average, we found our implementation performed worse than the implementation in Lui [1], which could be the result of two factors; the first is the inability to modify the data into a rotationally and reflectionally invariant form, as discussed previously, impeding the ability to perform the logorithmic and exponential mappings and resulting in different geometrical properties of the video space being analyzed; and the second is that several of the datasets analyzed by Lui [1] and Lui et al. [16] utilize preprocessing that is not described in detail, and thus we are unable to recreate it. The primary challenges of this project were acquiring an understanding of the geometry and manifold properties described in the literature, and acquiring sufficient understanding of topology and tensor decomposition to perform the required analysis. The majority of our struggles resulted from attempting to transform that knowledge into code, especially with regards to transforming the Grassmann manifold. With few libraries and resources at our disposal for working with manifolds, we were forced to manually code in much of the geometry of the tensors, resulting in difficulty keeping code between authors and datasets compatible and leading to many slow development periods. The advantages of this regression method for classification are primarily in the low-complexity and closed-form nature of training; this results in no hyperparameters and a relatively low computational time compared to deep learning methods. However, the closed-form least-squares formulation presented in Wang et al. [18], while potentially more optimal than the Karcher mean used in Lui [1], suffers from poor scalability, as it becomes exponentially slower to calculate as the video dataset grows in size, frame count, and resolution, and as such it is much better suited to smaller, less-complex datasets. The authors observed a relative lack of literature regarding the manifold geometry of video spaces in recent years, and believe that the field may have a great deal of untapped potential; in particular, much of the attention in modern video analysis and classification seems to be oriented more towards deep learning methodologies.

9 Contribution by Group Members

Author 1 and Author 2 take the primary workload in the actual coding. Author 1 and 2 both worked on pre-processing the different data sets into a tensor representation of videos. Author 1 implemented the closed form of Karcher mean and implemented it on top of the 1NN classifier. Author 2 implemented the 1NN classifier including the process of factorizing tensors into manifolds and providing implementations of the necessary mathematical operations on Manifolds. Author 3 works mainly on assisting on programming, finding relevant sources and data sets for implementation, and assisting in data architecture compatibility between code from Authors 1 and 2 . All three Authors participate in writing the Project Proposal and the Final Project Report.

References

[1] Yui Man Lui. Human gesture recognition on product manifolds. *Journal of Machine Learning Research*, 13(106):3297–3321, 2012. URL `http://jmlr.org/papers/v13/lui12a.html`.

[2] Antoni Chan and Nuno Vasconcelos. Probabilistic kernels for the classification of auto-regressive visual processes. 1:846– 851 vol. 1, 07 2005. doi: 10.1109/CVPR.2005.279.

[3] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (in conjunction with ICCV)*, 2005.

[4] H. Wang, M. Ullah, A Klaser, I. Laptev, and C. Schmid. Evaulation of local spatio-temporal features for action recognition. *British Machine Vision Conference*, 2009.

[5] P. Bilinski and F. Bremond. Evaluation of local descriptors for action recognition in videos. *ICVS*, 2011.

[6] P. Saisan, G. Doretto, Y-N. Wu, and S. Soatto. Dynamic texture recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[7] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto. Recognition of human gaits. *In IEEE Conference on Computer Vision and Pattern Recognition, Hawaii*, pages 270–277, 2011.

[8] A. Veeraraghavan, A. K. Roy-Chowdhury, and R. Chellappa. Matching shape sequences in video with applications in human movement analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (12):1896–1909, 2005.

[9] D. Kendall. Shape manifolds, procrustean metrics and complex projective spaces. *Bull. London Math. Soc.*, (16):81–121, 1984.

[10] P. Turaga and R. Chellappa. Locally time-invariant models of human activities using trajectories on the grassmannian. *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[11] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104:249–257, 2006.

[12] M. Rodriguez, J. Ahmed, and M. Shah. Action mach: A spatio-temporal maximum average correlation height filter for action recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[13] A. Datta, Y. Sheikh, and T. Kanade. Modeling the product manifold of posture and motion. *Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (in conjunction with ICCV)*, 2009.

[14] Z. Lin, Z. Jiang, , and L. Davis. Recognizing actions by shape-motion prototype trees. *IEEE International Conference on Computer Vision,*, 2009.

[15] R. Li and R. Chellappa. Group motion segmentation using a spatio-temporal driving force model. *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[16] Yui Man Lui, J. Ross Beveridge, and Michael Kirby. Action classification on product manifolds. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 833–839, 2010.

[17] Agnieszka Michalczuk, Kamil Wereszczy´nski, Jakub Segen, Henryk Josi´nski, Konrad Wojciechowski, Artur Bak, Slawomir Wojciechowski1, Aldona Drabik1, and Marek Kulbacki. Manifold methods for action recognition. pages 613–622, 2017. doi: 10.1007/978-3-319-54430-459.

[18] Yuping Wang, Lichun Wang, Dehui Kong, and Baocai Yin. Extrinsic least squares regression with closed-form solution on product grassmann manifold for video-based recognition. *Mathematical Problems in Engineering*, 2018:6598025, Mar 2018. ISSN 1024-123X. doi: 10.1155/2018/6598025. URL `https://doi.org/10.1155/2018/6598025`.

[19] Jose M. Chaquet, Enrique J. Carmona, and Antonio Fernández-Caballero. A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding*, 117(6):633–659, 2013. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2013.01.013. URL `https://www.sciencedirect.com/science/article/pii/S1077314213000295`.

[20] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007.