Context-Sensitive Human Activity Classification in Collaborative Learning Environments

Abigail Ruth Jacoby[1], Marios S. Pattichis[1], Sylvia Celedón-Pattichis[2] and Carlos LópezLeiva[2]
abby.jacoby@gmail.com, {pattichi,sceledon, callopez}@unm.edu
[1] image and video Processing and Communications Lab (ivpcl.unm.edu)
Dept. of Electrical and Computer Engineering
University of New Mexico, United States.
[2] Dept. of Language, Literacy, and Sociocultural Studies
University of New Mexico, United States.

***Abstract*—Human activity classification remains challenging due to the strong need to eliminate structural noise, the multitude of possible activities, and the strong variations in video acquisition. The current paper explores the study of human activity classification in a collaborative learning environment.**

This paper explores the use of color based object detection in conjunction with contextualization of object interaction to isolate motion vectors specific to each human activity. The basic approach is to make use of separate classifiers for each activity. Here, we consider the detection of typing, writing, and talking activities in raw videos.

The method was tested using 43 uncropped video clips with 620 video frames for writing, 1050 for typing, and 1755 frames for talking. Using simple KNN classifiers, the method gave accuracies of 72.6% for writing, 71% for typing and 84.6% for talking. Classification accuracy improved to 92.5% (writing), 82.5% (typing) and 99.7% (talking) with the use of Deep Neural Networks.

***Index Terms*—human activity classification; context-based methods.**

I. Introduction

Human activity recognition in video presents significant challenges. Many issues stem from the wide variety of identifiable human activities which may appear concurrently. There is clearly no established method for handling arbitrary human activities.

We begin with a short summary of recent research in human activity recognition. In [1], the authors used a weakly supervised Recursive Neural Net (RNN) with a probabilistic inference model to identify human activity over extended sequences. In [2], the authors used RGB color model with optical flow features to train a Long Short Term Memory Network (LSTM) to achieve 87% accuracy on the UFC101 dataset. In [3], the use of different neural networks for the same problem was explored. Within the same study, they showed that the use of optical flow did not have a significant impact on their approach. In [4], the authors investigated methods for accelerating the computations. In [5], the authors introduced the use of procedural neural networks (ProcNets) as a weakly supervised approach to learning based on temporal alignment. More recent studies are based on LSTM as reported in [6] and [7].

Fig. 1: Human Activity Recognition in Collaborative Learning Environments. In the example, we have a typing and a talking activity.

For the purposes of this paper, we are interested in understanding how students learn programming in a collaborative learning environment (see Fig. 1). We identified talking, writing, and typing as the primary activities of interest. Here, we are strongly interested in listening to students talk about coding, understanding how much time they spent typing their code, and monitoring the amount of time spent working with pencil and paper.

Our proposed approach is to consider the contextualized interaction of a collection of objects. Thus, in our approach, writing requires the detection of the pencil, the paper, and then the motion vectors of the pencil on the paper. Similarly, typing requires the detection of the keyboard, the human fingers on the keyboard, and the motion vectors associated with typing. Once all of the candidate objects and associated motions have been detected, we form a feature vector that is then fed to a classifier for identifying the activity. While our current paper is not concerned with fast implementations, for future work in hardware acceleration, we refer to our recent work in [8],

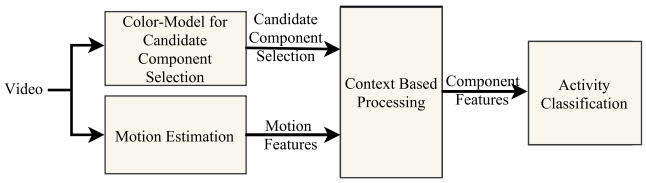

Fig. 2: General diagram of the human activity detection method.

[9] for implementing fast convolutions and cross-correlations using scalable architectures.

The remainder of this paper is organized into three sections. In section II, we provide a detailed description of the proposed method. We summarize our results in section III and provide concluding remarks in section IV.

II. Methodology

We present an overview of the method in Fig. 2. Initially, we apply color-based segmentation to extract the candidate objects of interest. For each video frame, we extract motion vectors that are specific to the objects of interest. We then apply context-based rules to filter and identify components that can be associated with specific activities (see Table I). For the specific components of interest, we extract motion vector features that are used for activity classification. In what follows, we describe the various methods that are involved.

A. HSV Color Models for Pencils, Table, Paper, Keyboard, and Faces

The goal of our use of the color models is to determine candidate components that are further processed based on their relative context. Thus, our use of color models produces an over-segmentation of the objects of interest. Bound selection was done visually using a simple database of 20 examples of pencils, paper, tables, skin regions and keyboards. Here, we found thresholds that worked on all images at the same time, as verified in the visual display of all examples.

B. Context-based Processing and Feature Extraction

The color models provide candidate regions for further processing. The candidate regions need to be carefully selected and then processed for context by processing relations between them. Then, a combination of checks is applied to check for interactions between them and motion content. The histograms of motion magnitudes and orientations are then used as features for further classification. In what follows, we provide more details for each step.

C. KNN Classifier for Selecting Keyboard and Face Components

For keyboard and face detection, we apply morphological filtering to remove minor regions. We then compute the bounding box for each region, zero-pad, convert to grayscale (Y component in Y-Cr-Cb), and use bilinear interpolation to resize each one of them to 128x128. We then use K-nearest neighbor to classify each component (e.g., keyboard present or not). The faces KNN classifier is trained using a collection of 1,700 images from various videos within our datasets, and labeled 0 or 1 to identify faces versus not faces respectively. For the keyboard, mouse, and monitor, we used 170 images.

D. Writing

We summarize context-based feature extraction for the writing activity in flowchart format in Fig. 3. Parameter optimization was performed through a visual interface.

For writing, we first use the binary image created by masking the table and take the bitwise and between the images to preserve only the objects on the table. This mask is applied to both the paper and the pencil binary images. We remove detections with unrealistic aspect ratios (as determined by a small training set). We then check that the pencil is on top of a table and over top of a paper which indicates writing may be present. The combination of color and aspect ratio for selection achieve tolerance to rotation and scaling.

The final step is to check for motion within the region of interest using the flow passed to the contextualization function. If a certain motion magnitude threshold is exceeded, we then extract the motion vectors from this region of interest and calculate a histogram from them. Histograms are computed per component and returned by the function. These features are later used in determining a classification of the motion as either writing or not writing.

E. Typing

We provide a flowchart summary of the typing activity algorithm in Fig. 4 . In what follows, we explain each step.

Typing follows a similar identification method as for pencil detection, with exception of a necessary extra step for elimination of large gaps between the keyboard and the monitor. To overcome this, we calculate the convex hull from the contour of the table, and apply a bitwise and with the resulting threshold image from the hull to close holes within the table.

The contours are found for each of the objects in the keyboard threshold, and after the centroids are located, we use the ROI to first resize the slice to 128x128 and fit the cropped grayscale image to a KNN model trained to classify keyboards, monitors and mice.

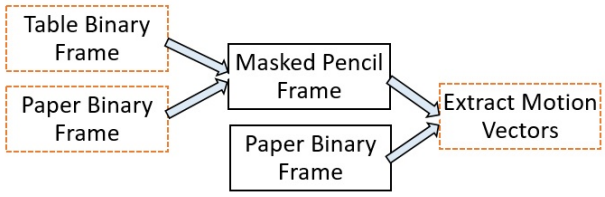

Fig. 3: Flowchart for confirmation of a pencil's precense prior to motion vector extraction.

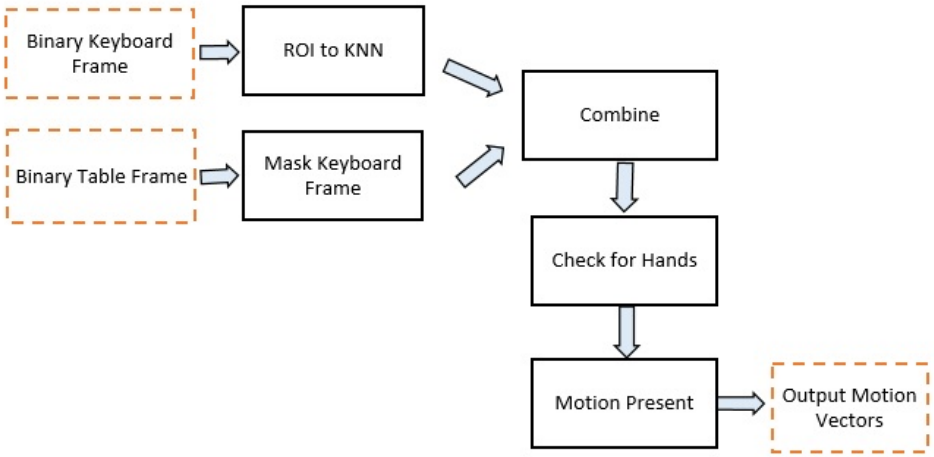

Fig. 4: A flowchart representing the steps to extracting the keyboard and discovery of potential typing.

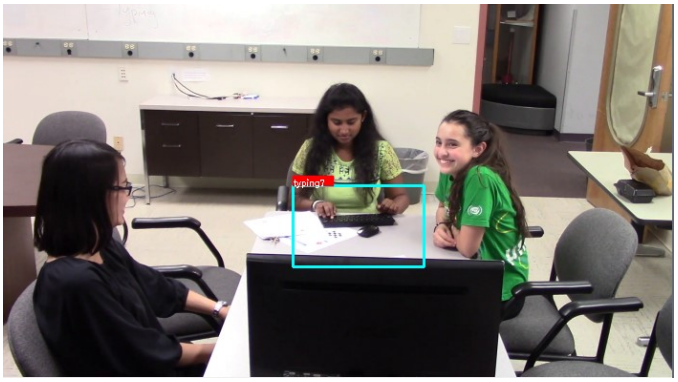

Fig. 5: A keyboard which has been properly found and classified for motion vector extraction.

If the object is determined to be a keyboard by the model, we then do a check for hands by slicing the skin image within the ROI box and checking to see if the remaining pixel value sum is greater than zero. If it is, we then calculate the appended histogram of the magnitude and phase of the optical flow as done previously. Again we visually review the resulting ROI boxes as an informal optimization of the parameters of each of our contexts.

F. Talking

To identify talking, the steps are slightly different. Regions of skin are found using the same technique as for the pencil. Some additional processing is done in this case to expand the areas around the extracted region using a gaussian blur before masking the frame, this includes also using a morphological open operation to remove noise from the similar colored bookshelves in the background. The blurring extrapolates a larger portion of the image around the skin area so the contour for evaluation in the KNN is more likely to contain the full face.

Using the threshold of this masked image, each contour is extracted and the boxed region about the centroid is resized to 128x128, as with the keyboard, and passed to a KNN model. If the area is matched to being a face, we apply the golden ratio and check only the bottom portion of the face by dividing the height by 1.618 and adding to the y value to get the top y-location of the box.

G. Feature vectors

The feature vectors are extracted every 3 frames as given in Table I. Specifically, we calculate probability density functions (PDFs) of the magnitude and angle from the motion vectors and append them together. Each set of PDFs are separated by the centroid coordinates.

H. Classification

We classify feature vectors for object presence. We investigated the use of KNN and the use of fully-connected deep neural nets (DNN). Different classifiers were considered as described in the results.

	Writing	**Typing**	**Talking**
Object Check	40<area<2000px and aspect ratio >1.2 or <0.5	200<area<8000px and Has 3+ corners	area>1000px and0.5≥aspect ratio≥1.5
Context Check 1	Pencil within table region	Keyboard detected on table	Face is detected
Context Check 2	Pencil near a piece of paper	Hands inside keyboard bounding box	Bottom part of face has motion
KNN Used	Yes	Yes	Color Only

TABLE I: Context Conditions for Object Recognition.

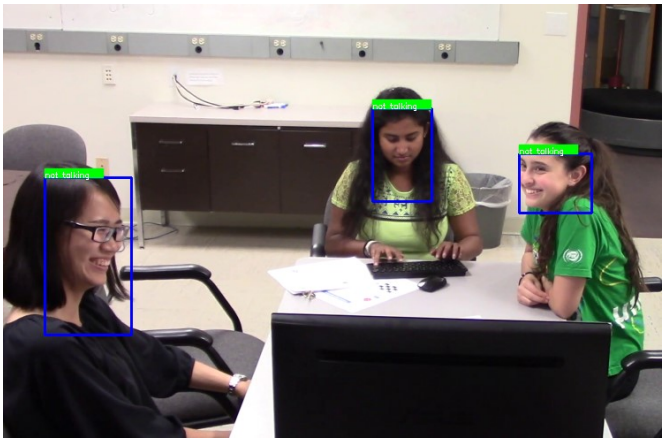

Fig. 6: Talking classifier example. In this example, the classifier correctly identified the faces. Then, based on the extracted motion vectors over the mouth area, a second classifier identified who is talking and who is not talking.

III. Results

We summarize the dataset in Table II. To select the best parameters for each method, we used nested cross-validation. Within the training set, we perform five-fold randomized cross-validation for optimal parameter selection. Final results are then reported on a tenfold cross validation of the selected model on a tenth of the database that was not seen during the training phase.

For KNN, we optimize the hyperparameters of distance and neighbors using ranges of $K = 3$ to $K = 19$, and explore the use of both Euclidean and city-block distance for determining the nearest neighbor. Additionally, we explore the use of feature scaling for motion vector analysis. Using simple KNN classification, the method gave accuracies of 72.6% for writing, 71% for typing and 84.6% for talking.

Classification accuracy improved to 92.5% (writing), 82.5% (typing) and 99.7% (talking) with the use of trained Deep Neural Network classifiers. We carefully document the optimal DNN architectures in Table III. We note that the optimal KNN classifiers were significantly less accurate than DNN.

From the results, it is clear that the talking classifier performed very well. An example of correct talking classification is shown in Fig. 6. Similarly, at 92.47%, the accuracy for the writing classifier also worked well. The accuracy of the typing classifier was lower at 82.52%. Overall, the use of multiple classifiers provided for careful selection of the candidate activity regions. Careful selection of color models with added KNN classification for object detection greatly increase accuracy. The method must be altered to suit candidate objects under observation to perform correctly.

	Talking	**Typing**	**Writing**
No. of Features	1755	1050	620
No. of Videos	14	14	15
FPS	60	26-60	24-60
Duration (in Seconds)	5-16	5-24	1-39

TABLE II: Dataset information for the final training dataset.

	Talking	**Typing**	**Writing**
Accuracy	99.72%	82.52%	92.47%
Batch Size	100	100	200
Activation Model	Relu	Selu	Selu
Neurons	100	50	70
Learning Rate	0.01	0.05	0.05
Hidden Layers	5	5	5
Regularization	L1/L2 Max_Norm	L1/L2 Max_Norm	L1/L2 Max_Norm

TABLE III: Human activity classification using Deep Neural Networks. For each DNN classifier, we list the optimal network parameters.

IV. Conclusions

Overall, we have found that the proposed methodology worked well in most of our test videos. We are currently in the process of testing our methods in more challenging environments and over significantly larger datasets. Careful model selection for object detection greatly increases the accuracy of the method. We believe that the combination of context-based methods can be adapted to much wider applications.

V. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1613637 and Grant No. CNS-1422031.

References

[1] A. Richard, H. Kuehne, and J. Gall, "Weakly supervised action learning with rnn based fine-to-coarse modeling," 2017.

[2] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[3] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[4] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with enhanced motion vector cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2718–2726.

[5] L. Zhou, C. Xu, and J. J. Corso, "Procnets: Learning to segment procedures in untrimmed and unconstrained videos," *arXiv preprint arXiv:1703.09788*, 2017.

[6] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International Conference on Machine Learning*, 2015, pp. 843–852.

[7] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei, "Unsupervised learning of long-term motion dynamics for videos," 2017.

[8] C. Carranza, D. Llamocca, and M. Pattichis, "Fast and scalable computation of the forward and inverse discrete periodic radon transform," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 119–133.

[9] ——, "Fast 2d convolutions and cross-correlations using scalable architectures," *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2230–2245.