Context-Sensitive Human Activity Classification in Video Utilizing Object Recognition and Motion Estimation

Abigail R. Jacoby
University of New Mexico - Main Campus

Follow this and additional works at: http://digitalrepository.unm.edu/ece_etds

Part of the Electrical and Computer Engineering Commons

Recommended Citation

This Thesis is brought to you for free and open access by the Engineering ETDs at UNM Digital Repository. It has been accepted for inclusion in Electrical and Computer Engineering ETDs by an authorized administrator of UNM Digital Repository. For more information, please contact disc@unm.edu.
Abigail Jacoby

Candidate

Electrical and Computer Engineering

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Prof. Marios Pattichis, Chairperson

Prof. Manel Martinez-Ramon

Dr. Ramiro Jordan
Context-Sensitive Human Activity Classification in Video Utilizing Object Recognition and Motion Estimation

by

Abigail Ruth Jacoby

B.S., Computer Engineering, University of New Mexico, 2016

THESIS

Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science
Computer Engineering

The University of New Mexico
Albuquerque, New Mexico

May, 2018
Dedication

In memorium of my Aunt Linda, Uncles Gordon and Steve, and Grandmother Lois whom we’ve lost this semester, and of my Aunt Helen who passed 3 years recent.

To everyone who said I’ll never make it...

”I’ve always been a fighter. If you tell me I can’t, I’ll die trying to prove you wrong.”

– R. A. Salvatore
Acknowledgments

I thank my advisor and labmates for putting up with me this year, and my friends who have been indispensible to me while I was being an insufferable bore. Also to my cats, netflix, and actual gallons of coffee, I couldn’t have done it without you.

A special thank you is extended to the National Science Foundation for funding the AOLME project\footnote{This material is based upon work supported by the National Science Foundation under Grant No. 1613637 and NSF AWD CNS-1422031.} and for everything else they do to empower underrepresented groups in STEM.
Context-Sensitive Human Activity Classification in Video Utilizing Object Recognition and Motion Estimation

by

Abigail Ruth Jacoby

B.S., Computer Engineering, University of New Mexico, 2016
M.S., Computer Engineering, University of New Mexico, 2017

Abstract

Classification of human activity in raw video presents a challenging problem that remains unsolved, and is of great interest for large datasets. Though there have been several attempts at applying image processing techniques to video to recognize human activity in controlled video segments, few have attained a significant degree of success in raw videos.

Raw video classification exhibits significant challenges that can be addressed through the use of geometric information. Current techniques employ a combination of temporal information of the feature space or a combination of Convolutional and Recurrent Neural Networks (CNN and RNNs). CNNs are used for frame feature extraction and RNNs are then applied for motion vector extraction and classification. These techniques, which utilize information from the entirety of a frame, attempt to classify action based on all motion vectors and all objects found in the video. Such methods are cumbersome, often difficult to train, and do not generalize well beyond the dataset used.
This thesis explores the use of color based object detection in conjunction with contextualization of object interaction to isolate motion vectors specific to an activity sought within uncropped video. Feature extraction in this thesis differs significantly from other methods by using geometric relationships between objects to infer context. The approach avoids the need for video cropping or substantial preprocessing by significantly reducing the number of features analyzed in a single frame. 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 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 a trained Deep Neural Network.
Contents

List of Figures x

List of Tables xiv

Glossary xv

1 Introduction 1

1.1 Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.2 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

1.3 Thesis Statement . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1.4 Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2 Background 7

2.1 Classification of Human Activity in Video . . . . . . . . . . . . . . 13

3 Methodology 16
Contents

3.1 Overview of the Approach ................................................. 16
3.2 Candidate Region Selection Based on Color Models ................. 17
    3.2.1 Color Models for Pencils, Table, Paper, Keyboard, and Faces 17
3.3 Context-based Processing and Feature Extraction ................. 21
    3.3.1 KNN Classifier for Selecting Keyboard and Face Components 21
    3.3.2 Writing .............................................................. 22
    3.3.3 Typing ............................................................... 28
    3.3.4 Talking ............................................................. 32
    3.3.5 Feature vectors .................................................... 36
3.4 Classification .............................................................. 36

4 Results .............................................................................. 37
4.1 Dataset ........................................................................... 37
4.2 Classification Model Results ............................................. 38
    4.2.1 Results for Writing .................................................. 38
    4.2.2 Results for Typing ................................................... 40
    4.2.3 Results for Talking .................................................. 43
4.3 Failed Classification Examples ........................................ 45
4.4 Discussion ...................................................................... 52

5 Conclusions and Future Work ............................................. 56
## List of Figures

1.1 Writing (a) v.s Not Writing (b), Typing (c) v.s. Not Typing (d), and Talking (e) v.s. Not Talking (f) .......................... 3

1.2 Cropped video sample. ................................................. 4

3.1 Overview flowchart. .................................................. 16

3.2 Masked pencil color (right) v.s. original frame (left). ............. 18

3.3 Subfigure a shows the collection of pencil variations we see, and b shows what we wish to achieve after masking and thresholding. . . 18

3.4 A cropped writing video frame with potential pencil highlighted. . 19

3.5 Flowchart displaying the process of extracting colors from a frame. . 20

3.6 A sample of faces that are used to train the face KNN model. .................. 22

3.7 A collection of ”Non-Faces” used in training the face KNN model. . 22

3.8 A collection of figures used in training the keyboard KNN, with keyboard samples in (3.8a),(3.8b), and (3.8c), and non-keyboards (monitors in in (3.8d), (3.8e) and mice in (3.8f), (3.8g), (3.8h)) ....... 24

3.9 A correctly identified keyboard using KNN .......................... 25
List of Figures

3.10 A correctly identified face in KNN. ........................................ 25
3.11 Paper detection by color, intermediate steps. ......................... 26
3.12 Pencil detection by color, intermediate steps. ....................... 26
3.13 Flowchart for confirmation of a pencil’s presence prior to motion vector extraction. .................................................. 27
3.14 Pseudocode for function Context() part 1, writing. ............... 28
3.15 Images used as input for masking the background of the keyboard image ................................................................. 29
3.16 Figures (a) and (b) display the two components required to determine if typing may be present. .............................. 29
3.17 A keyboard which has been properly found and classified for motion vector extraction. ................................................. 30
3.18 A flowchart representing the steps to extracting the keyboard and discovery of potential typing. ................................. 31
3.19 Definition for function Context() part 2, typing. ................. 32
3.20 Correctly identified faces (left) and skin regions that were sent to KNN (right). ............................................................. 33
3.21 Process of selecting a region for KNN and the output. .......... 34
3.22 A flowchart representing the steps to identifying a face. ........ 35
3.23 Pseudocode for function Context() part 3, talking. ............ 35
4.1 Subfigures and display correct classification results for the color and context model of finding a pencil. ............................... 38
List of Figures

4.2 A correctly classified writing activity. ........................................... 39
4.3 Results in an outside unseen video from the AOLME database. ... 39
4.4 Results from model in an outside unseen video from the AOLME dataset. ................................................................. 40
4.5 Correct keyboard identification. ................................................... 40
4.6 Correct classification in outside unseen video from AOLME dataset. 41
4.7 Correct classification of typing activity. ................................. 41
4.8 Correct classification of not typing in difficult case. ...................... 42
4.9 Correct classification of typing activity in difficult case. ............... 42
4.10 Subfigures and display correct classification results for the color and correct KNN results for the face. ................................. 43
4.11 Correct classification of face KNN during validation. .................. 43
4.12 Subfigures and display correct classification results for no talking and talking. .......................................................... 44
4.13 Correct classification of talking activity in difficult case in unseen video. ................................................................. 45
4.14 Incorrect classification of table and legs as keyboard. .................... 46
4.15 Subfigures and display incorrect functionality of the face KNN from skin and blank space. ................................................ 46
4.16 Classification of a mouse as a keyboard. ................................. 47
4.17 Table masking issue. ................................................................. 47
List of Figures

4.18 Incorrect masking due to whiteboard. .......................... 48
4.19 Keyboard classified as mouse. ................................. 48
4.20 Face missed due to hand on face. ............................ 49
4.21 Subfigures displaying incorrect classifications from all models. .... 50
4.22 Confusion matrices for object KNN. Mon stands for Monitor. .... 51
4.23 Confusion matrices for KNN Results. .......................... 52
List of Tables

2.1 RGB+Optical Flow Methods ........................................... 8
2.2 Temporal RNN Methods .................................................. 10
2.3 Commonly Used Datasets ............................................... 11

3.1 Context Conditions for Object Recognition ......................... 36

4.1 Dataset Video Types ..................................................... 37
4.2 Dataset information for the final training dataset. ............... 38
4.3 KNN Models for Typing and Talking ................................. 51
4.4 KNN Results .............................................................. 51
4.5 Tensor-flow KNN Results ............................................... 51
4.6 DNN Results .............................................................. 52
# Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOLME</td>
<td>The Advancing Out of School Learning in Mathematics and Engineering research study.</td>
</tr>
<tr>
<td>SotA</td>
<td>State of the Art.</td>
</tr>
<tr>
<td>Binary Image</td>
<td>An image consisting only of black and white values.</td>
</tr>
<tr>
<td>Centroid</td>
<td>The center point in a contour.</td>
</tr>
<tr>
<td>Contour</td>
<td>The collection of points representing the outline of a 'blob' in a binary image.</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network.</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Network.</td>
</tr>
<tr>
<td>Feature Space</td>
<td>The n-dimensional space in which the variables to be classified by a machine learning algorithm to be classified are stored.</td>
</tr>
<tr>
<td>Ground Truth</td>
<td>A set of labelled data that serves as a point of comparison for classification accuracy.</td>
</tr>
<tr>
<td>HSV</td>
<td>Hue, Saturation, Value color model.</td>
</tr>
</tbody>
</table>
### Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN</td>
<td>K-nearest neighbors, a machine learning model that selects a classification for an object based on the distance between the object of interest and its k closest neighbors in a graph.</td>
</tr>
<tr>
<td>LTSM</td>
<td>Long Short-Term Memory, a block of a neural network that serves as a building component, i.e. short term memory which can extend over long periods of time when many are layered together.</td>
</tr>
<tr>
<td>LCRN</td>
<td>Long-term Recurrent Convolutional Networks.</td>
</tr>
<tr>
<td>ProcNet</td>
<td>A Procedural Neural Network.</td>
</tr>
<tr>
<td>RGB</td>
<td>Red, Green, Blue; referring to a color image frame in a video.</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network.</td>
</tr>
<tr>
<td>ROI</td>
<td>Region of Interest.</td>
</tr>
<tr>
<td>STEM</td>
<td>Science, Technology, Engineering and Mathematics.</td>
</tr>
<tr>
<td>Video in the Wild</td>
<td>Video recorded in an uncontrolled environment.</td>
</tr>
<tr>
<td>YCrCb</td>
<td>Y for Luminance, Cb for Chrominance-Blue, and Cr for Chrominance-Red. Another color value model.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Overview

Human activity recognition in raw video is a challenging problem that has drawn a lot of interest, especially around classification tasks involving huge video datasets. Most current methods have focused on speeding-up previous approaches, methods by which would otherwise require an impossible amount of human time to manually classify. However, these classification methods, even when assisted by humans, are not yet designed to accurately handle video in the wild.

An example of a large video database (several TB worth of raw video at the time of this writing) is the AOLME video database. The dataset collected for the AOLME project is an agglomeration of information in raw video that is currently being analyzed manually. The data were collected for the purposes of understanding how children learn in situations involving mathematical and programming challenges, such that teaching methods can be improved to broaden underrepresented student participation in STEM fields. The information necessary for identifying key features of the video include the discovery of interactions between the students and their
Chapter 1. Introduction

facilitators for the purposes of identifying best teaching practices.

Because of the sheer size of the dataset, much of the video will never be fully analyzed. This being, in part, due to the many hours necessary to manually classify each video for each individual activity. In these cases, clearly an efficient, accurate classification tool to aid in the discovery for those researchers such that they could quickly identify videos bearing certain activities is warranted and could greatly improve the quality of their results.

1.2 Motivation

The majority of current human activity detection methods require a significant amount of human time to clip out regions of interest and train a system to correctly identify actions within each region. Referred to as the ground truth, this manual labeling of cropped video segments has been used for training for the majority of machine learning classifiers to date. The current thesis focuses on identifying multiple video activities in raw videos without the requirement for manual cropping individual video segments prior to automatic classification.
Chapter 1. Introduction

Figure 1.1: Writing (a) v.s Not Writing (b), Typing (c) v.s. Not Typing (d), and Talking (e) v.s. Not Talking (f)

The thesis focuses on the detection of human video activities that are shown in Figure 1.1. More specifically, the thesis will consider: writing vs not writing as seen in Fig. 1.1a and 1.1b, typing versus not typing as seen in 1.1c and 1.1d, and finally talking versus not talking in 1.1e and 1.1f within a full, uncropped video. In comparison, more conventional methods would consider the same detection problems where the video activity has been isolated as demonstrated in Fig. 1.2. On the other hand, for training purposes, the thesis will consider training on the cropped video activities. To differentiate between the two database types, we refer to the original
Chapter 1. Introduction

videos as raw videos as opposed to the cropped videos.

![Cropped video sample.](image)

Figure 1.2: Cropped video sample.

The proposed research presents a novel approach to classification of video activity which can classify human activity with regards to the contextualized interaction of a collection of objects which have been identified using color. The method attempts to better classify human activity by considering the statistics of motion vectors extracted from regions of interest associated with adjacent objects. The approach is different from the standard methods that classify human activity through the use of full-frame methods that cannot be associated with specific objects or motions.

1.3 Thesis Statement

The thesis of this research states that a specific human activity of interest can be classified in raw video given a combined set of objects identified by means of color, motion vectors about the primary object, and the use of contextualization algorithms for each sought activity. The research of this thesis assigns a predicted classifica-
Chapter 1. Introduction

tion based on the interactions between the objects, referred to as giving context to an interaction. A final classification may then be assigned using machine learning techniques to examine motion vectors within the region of interest which has been identified. This thesis is focused on the determination that the proposed method which contextualizes object recognition prior to analyzing motion vectors is a viable solution for detecting specific individual activities in raw video, and additionally within other videos unseen by training. Furthermore, this thesis states the provided method is easily adjustable for identification of activity given various other human-object interactions by their associated context, and that activity recognition can still be ascertained given alternate methods of object detection.

1.4 Contributions

The primary contribution of this thesis is to provide a new method for determining human activity based on the interactions of detected objects in raw videos. Mainly, the contribution is that the thesis provides a method for a dramatic reduction in the number of features in activity recognition. The thesis adds locality to the many RGB+flow methods that are currently considered among the most successful classification efforts, allowing context to be applied to the exact activity being sought. Human activities restricted to small regions of interest that were not previously classifiable by other methods can be analyzed with the proposed method (e.g., differentiating between a student playing with a pencil and writing).

1.5 Summary

The remainder of this thesis is divided into five chapters. In Chapter 2 we explore the current state of the art methods for activity recognition in video, along with
Chapter 1. Introduction

their respective datasets, and supply ample comparison to our algorithm. Chapter 3 explores the proposed algorithm in detail, beginning with an overview of the full model and breaking down the method to individual algorithms. A presentation of the findings are displayed visually, summarized into tables, and briefly discussed in Chapter 4. Finally, Chapter 5 continues with a summary discussion of the results and closes with conclusions and suggested future work. An appendix of code is included in Chapter 6 for reference.
Chapter 2

Background

Video activity classification is an expansive area of study which yearly generates an enormous amount of interest. Currently there is no generalizable solution, thus there have been multiple attempts to solve the problem in a way that can be applied outside of training datasets. Of particular interest are effective algorithms to segment video that has not been preprocessed in any way for classification of activity. This is made difficult by the numerous, transient human activities that may take place in any single video because they cannot be readily separated locally, and sometimes may be recognized as a motion within a sea of noise. Confounding issues further, with classified models trained on small clips of single activities of interest, an extremely computationally expensive comparison of the entire feature space of a set of vectors to match the trained template is necessary. Again, this fails in the case that an activity is associated with motions over small regions of interest, though the very existence of the sought activity could be ascertained with the human eye.

The state of the art explored in the literature review for this thesis focused heavily on methods of temporal frame differencing, bag of features, RNNs to recursively learn annotated video, optical flow methods for motion estimation, CNNs for feature
## Chapter 2. Background

<table>
<thead>
<tr>
<th>Study</th>
<th>Dataset</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Stream Convolutional Networks for Action Recognition in Videos, 2014 [274]</td>
<td>UCF-101 and HMDB-51</td>
<td>Combines RGB frame and flow inputs for frame by frame recognition on a new network combination. 87.0% on UFC-101, 59.4% on HMDB-51</td>
</tr>
<tr>
<td>Beyond Short Snippets: Deep Networks for Video Classification, 2015 [298]</td>
<td>Sports 1 million dataset, and the UCF-101 datasets with and without additional optical flow information</td>
<td>Improved dense trajectory motion estimation by removing camera movement by (73.1% vs. 60.9%), (88.6% vs. 88.0%), and (82.6% vs. 73.0%), respectively</td>
</tr>
<tr>
<td>Long-term Recurrent Convolutional Networks for Visual Recognition and Description, 2016 [56]</td>
<td>UCF101 dataset, AlexNet, CaffeNet</td>
<td>LCRN builds on [274], applies models to conventional activity challenges. Averages all frames to classify, RGB: 68.20%, Flow: 77.28%</td>
</tr>
<tr>
<td>Real-time Action Recognition with Enhanced Motion Vector CNNs, 2016 [299]</td>
<td>UFC101, ImageNet and ILSVRC-2012</td>
<td>Aim to speed up the RGB-Optical flow method [274] for use with realtime video classification 61.5% 403.2fps</td>
</tr>
</tbody>
</table>

Table 2.1: RGB+Optical Flow Methods

extraction from a single frame along with frame differencing for motion (flow), or some combination of the above. A summary of current research is provided in Table 2.1. All methods pertain to frame feature classification and optical flow.

Many of the attempts to classify video actions are now using two or more trained neural networks to make their predictions. The methods pass each frame for feature extraction through a CNN then form a list of objects, combine with motion estimated
Chapter 2. Background

through an RNN or other type of recursive learning network, typically an LTSM method, then combine the two outputs to make a prediction for the activity based on learning models or probabilities. One such attempt passed a list of all the activities the video contained along with the video to train a weakly supervised RNN, making use of youtube annotations [261].

A method similar to the proposed method feeds RGB data to one trained network and optical flow data to another finally combining them with an LTSM achieving 87% accuracy on the UFC101 dataset [274]. This research is used as the basis for many of the following bodies of research. They allow a CNN to classify all objects in training frames prior to sending the motion vectors for classification, then after having a classification for every frame give the action a classification.

An example study that used different combinations of neural networks for this type of classification, such as LTSM instead of RNN, was done by [298] and barely improved upon the previous method [274]. Their model attempted to classify a video with only raw frames as well. Their results are interesting to us as when they removed optical flow they saw only a minor decrease in accuracy which could indicate either overfitting or the probability models being independent from one another.

More recently a study that builds from [274] simply attempted to speed up the framerate so that it could be applied in realtime as opposed to waiting for all frames to be estimated [299]. Some of these models such as in [298] sought ways to improve the motion vector classification, also seen in [291]. We begin to see a pattern in the state of the art methods where followup methods do not attempt to compete but to improve upon the base RGB+flow method.

Methods developed within the last year are looking more toward the use of LTSM to identify sequences of frames over short time-spans. These are referred to mostly as temporal frame extraction techniques with segment procedural learning and require
Chapter 2. Background

<table>
<thead>
<tr>
<th>Study</th>
<th>Dataset</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recurrent Assistance: Cross-Dataset Training of LSTMs on Kitchen Tasks, 2017 [249]</td>
<td>Breakfast (18.8), Salads (46.86%), and MPII Cooking 2 (41.1%)</td>
<td>68.5% and 94.0% on HMDB51 and UCF101 respectively, 18.59% average obtained by the (best performing) RGB stream in their model</td>
</tr>
<tr>
<td>Towards Automatic Learning of Procedures from Web Instructional Videos, 2017 [300]</td>
<td>A series of instructional cooking videos</td>
<td>Segment/label a video and train a ProcNet to recognize each frame based on previous labels; 30.4% precision, 37.1% recall</td>
</tr>
<tr>
<td>Weakly Supervised Action Learning with RNN based Fine-to-coarse Modeling, 2017 [261]</td>
<td>Breakfast dataset and Hollywood extended dataset</td>
<td>Provide a list of activities and allow RNN to classify each activity using feature extraction Accuracy 33.3%, SotA: 27.7%</td>
</tr>
</tbody>
</table>

Table 2.2: Temporal RNN Methods

an incredible amount of time training machines or large annotated youtube datasets which can be referenced quickly from Table 2.3. We summarize these works in Table 2.2.

One such study provided an already labeled activity and used an RNN to learn the activity using objects that were found in the frame, somewhat of a reverse of the previous case [261]. The paper gave an example of a hand, teabag, and teacup indicated somebody was making tea. Another method used what they referred to as ProcNets, or procedural neural networks, as a weakly supervised approach to learning based on temporal alightment [300]. Here, they use a similar method which used previous information to determine if the estimate ”made sense”. The example given was that it isn’t likely that someone is mixing dough if they cracked an egg
and then two frames later are cracking an egg [249]. Several more studies of LTSM only approaches both supervised and unsupervised have been reported in [279] and [137].

<table>
<thead>
<tr>
<th>Study</th>
<th>Dataset</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hollywood</td>
<td>Extended Dataset</td>
<td>937 videos with 16 different actions from 69 movies.</td>
</tr>
<tr>
<td>MPII Cooking 2</td>
<td></td>
<td>87 unique cooking activities with annotations.</td>
</tr>
<tr>
<td>Breakfast</td>
<td>Dataset</td>
<td>A video dataset containing video of 10 common breakfast chores along with annotation.</td>
</tr>
<tr>
<td>UCF-101</td>
<td></td>
<td>101 human action classes with over 13,000 video clips.</td>
</tr>
<tr>
<td>HMDB-51</td>
<td></td>
<td>A human motion dataset consisting of 51 human actions.</td>
</tr>
<tr>
<td>ImageNet</td>
<td></td>
<td>A large image database specifically for image software research projects.</td>
</tr>
<tr>
<td>YouTube-8M</td>
<td></td>
<td>YouTube video IDs and labels from 4700+ entities.</td>
</tr>
<tr>
<td>YouCook</td>
<td>II</td>
<td>88 cooking videos from youtube.</td>
</tr>
</tbody>
</table>

Table 2.3: Commonly Used Datasets

The classic approach to training consists of segmenting videos into short clips which are pre-processed, labeled, and then used as input to a machine learning model. The fit data is then tested using other clips showing the same activity. These video clips contain typically only one person performing one action, or are sent after a
Chapter 2. Background

machine has been trained with a series of labeled images; there is a massive amount of object training that must be done in either situation. Training a cascade to recognize a new object is cumbersome and can take quite a bit of time, and sometimes will not be enough to recognize the object of interest. Imagenet and other repositories have readily available feature cascades model zoos which are quite limited in terms of content, or objects, they are trained to recognize; though, they are rapidly becoming more diverse. Currently trained object recognition cascades or machines are few and far between, sometimes not publicly available, and do not necessarily detect objects related to education, such as a pencil, pen, eraser, or other such writing implements. These objects are of particular interest in the AOLME video library used in the testing of this model, so having a way to detect writing implements accurately via object detection would have been of great value to this research.

Further, review discovered that most all studies used the datasets presented in Table 2.3 for benchmarking the performance of each presented algorithm. These datasets are used as points of comparison for classification typically of objects of different human activities. Though these are the common datasets used, they do not have models of all objects under review, nor the particular activities being explored, and they do not provide the scale of data necessary to perform a robust review of the algorithm presented. The AOLME dataset contains several hours of human activity in the wild, rather than zoomed on a single action, and have many examples of the activities we wish to identify. This dataset has an inconsistent number of actions happening per frame, all of which need to be assigned context in a manner that the above datasets do not lend well to. Finally, the AOLME dataset has a consistent color scheme to the data provided (the objects used are all the same make/model), and as discussed, this algorithm relies on color to find objects, so for this thesis, a subset of this video database will be used to test the performance of the algorithm.

The AOLME project is an interdisciplinary collaboration between the UNM Col-
Chapter 2. Background

lege of Education and the School of Engineering (see [41]). Some of the earlier lessons learned in the AOLME project have been summarized in [247]. The AOLME dataset generated a large video dataset that presents several computing challenges (e.g., see similar problems in [197]). In research related to the current thesis, Cody Eilar looked at writing and typing classification using Amazon Web Services [57]. The recent emergence of convolutional neural networks is also of great interest to the ivPCL lab at UNM because of a long tradition in convolution-based methods (e.g., see [40], [37]) and image analysis methods (e.g., see [225], [237], [164], [156]).

2.1 Classification of Human Activity in Video

In review, many current ideas for activity recognition rely heavily on optical flow motion vectors to determine what is happening in the video clip. In a large video, where many activities could simultaneously be taking place, this creates an obvious problem of structural noise in the motion vectors, the obscurcation of which causes extreme difficulty when attempting to hone in on a specific motion. Additionally, calculating the feature vectors for an entire video is highly computationally expensive, and in attaining a good set of ground truth data often many different videos are required.

These issues lend to complicated neural networks being used to quickly facilitate the learning process. Neural Networks can have significant fallbacks as they are very prone to overfitting and in many cases perform as ”block boxes”, we simply cannot tell how they are coming to their conclusion or how they fail. This phenomenon is referred to as ”unexplainable AI”. In such systems, the problem is that the systems come to conclusions without being able to explain how the results were achieved. This, of course, is a significant problem.

Though some of the studies we reviewed inspected various approaches to overcome
Chapter 2. Background

these limitations (Temporal learning and speeding up of flow methods for realtime classification), they still suffer serious fallbacks. When applied to outside datasets, these methods tend to fail and do not generalize outside of their training dataset. Though our method also makes use of a neural network, it was not necessary to use a CNN or RNN to identify objects then learn features. Additionally, the features in this thesis are made very easy to visualize such that it is possible to inspect exactly where and how the algorithm fails.

There are many obstacles preventing generalization of activity recognition methods. High, structural noise motions, as previously mentioned, such as walking will occlude lower magnitude motions such as writing during background subtraction. This creates a problem of bleeding when creating blobs about which we are attempting to classify; the feature space then is nearly impossible to segment in a way such that each individual motion can be found, much less classified. This is further complicated by illumination variations, which can cause motion to be found in a video where there is no motion. This also has a great effect on motion vector analysis as high noise motions can drown out smaller motions. Geometric distortion and locality of an action in a full frame can also cause significant difficulty with classification. There are image techniques which classify a image’s scenet based on the relation of various objects found in its feature space [110]. We draw from this idea and attempt to generalize it for use in video by coming up with a geometrically invariant object interaction algorithm based on color. The proposed method removes the difficulty in determining an action’s existence in different datasets simply by first determining if an object of interest is present within a certain context. This may be a combination of objects, such as a plate, a fork, and an up and down motion, or simply an object itself, e.g. a pencil, being moved around by a human (which is made as an assumption since pencils cannot move on their own!)

More recently, researchers have been combining depth cameras to help identify
primary motion and colored object recognition. Since the AOLME dataset was recorded on a normal camera, we could not employ these methods and thus they were not pursued further. With regards to object recognition, this is currently done by using pre-trained CNNs or model zoos, such as AlexNet, to identify and classify features within a single frame. This is an exhaustively explored method of study with excellent reported accuracy on common datasets. Here, we note that we choose to implement the least computationally expensive method, recognition of objects using color. For the purposes of this research, the objects we wish to classify are primarily of uniform color. This thesis then follows a similar pattern to the more successful methods in that we will be combining object recognition with optical flow, though in a greatly reduced feature space.
Chapter 3

Methodology

3.1 Overview of the Approach

We present an overview of the method in 3.1. 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. For the specific components of interest, we extract motion vector
Chapter 3. Methodology

features that are used for activity classification. In what follows, we describe the various methods that are involved.

3.2 Candidate Region Selection Based on Color Models

3.2.1 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. We will later correct the over-segmentation result by eliminating invalid components based on context.

We develop a color-based segmentation system based on the HSV color space. Since we are working with single-color objects, the goal was to determine bounds for each component. 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. An example is shown in Fig. 3.2. The code for finding pencils can be found in Appendix A, specifically under A.1, and A.2.

Caution was taken to first visualize the selected colors in the video to confirm accurate detection. A ROI box was drawn on the frame around each centroid of each contour that remained after masking. Satisfied with a visual inspection over various videos from different datasets, we did not further optimize the values for colorspace. An example of the visual inspection is seen in Fig. 3.4. A flowchart summary of code is provided in Fig. 3.5.
Figure 3.2: Masked pencil color (right) v.s. original frame (left).

Figure 3.3: Subfigure a shows the collection of pencil variations we see, and b shows what we wish to achieve after masking and thresholding.
Chapter 3. Methodology

Figure 3.4: A cropped writing video frame with potential pencil highlighted.
Figure 3.5: Flowchart displaying the process of extracting colors from a frame.
3.3 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 details for each step.

3.3.1 KNN Classifier for Selecting Keyboard and Face Components

For keyboard and face detection, our use of the color models provides for initial candidate regions. Here, we found that these initial candidate regions needed to be further processed prior to use. 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). For the pre-KNN code, refer to A.3 for the keyboard, A.4 for the table, and A.5 for the skin detection. Figures 3.6 and 3.7 contain several examples of data used in the KNN training models.

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. Refer to section 3.4.
Chapter 3. Methodology

Figure 3.6: A sample of faces that are used to train the face KNN model.

Figure 3.7: A collection of ”Non-Faces” used in training the face KNN model.

3.3.2 Writing

We summarize context-based feature extraction for the writing activity in flowchart format in Fig. 3.13 and pseudocode format in Fig. 3.14. In what follows, we describe each step.

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.

The following step uses the binary image of the pencil as input to a find contours function in opencv, after which we use the moment of each contour and extract each centroid, and after obtaining the best fitting rectangle, calculate the aspect ratio of the contour. We check to make sure the aspect ratio and area are appropriate for a pencil, and then finally, using the centroid, we create a region of interest box around the pencil.
Chapter 3. Methodology

We use the dimensions of the region of interest box plus some padding to slice the paper binary image, effectively cutting out only areas near the pencil, and the sum of all the pixels remaining are taken. If the sum is greater than zero in this area, we determine that the pencil is on top of a table and over top of a paper, thus writing may be present.
Chapter 3. Methodology

Figure 3.8: A collection of figures used in training the keyboard KNN, with keyboard samples in (3.8a), (3.8b), and (3.8c), and non-keyboards (monitors in in (3.8d), (3.8e) and mice in (3.8f), (3.8g), (3.8h))
Chapter 3. Methodology

Figure 3.9: A correctly identified keyboard using KNN

Figure 3.10: A correctly identified face in KNN.
Chapter 3. Methodology

(a) Binary Paper Image  (b) Masked Paper Image

Figure 3.11: Paper detection by color, intermediate steps.

(a) Pencil Binary Image  (b) Masked Pencil Image

Figure 3.12: Pencil detection by color, intermediate steps.
Chapter 3. Methodology

Figure 3.13: Flowchart for confirmation of a pencil’s presence prior to motion vector extraction.

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, which is returned by the function and later used to determine whether the motion is due to writing or not.

Parameter optimization was performed through a visual interface. Here, the effect of the use of different parameter values is assessed through their impact on the ROI bounding boxes on a small set of training videos with different geometric angles, lighting conditions, and settings.
Chapter 3. Methodology

3.3.3 Typing

We provide a flowchart summary of the typing activity algorithm in Fig. 3.18 and provide pseudocode for the typing portion of function Context() in Fig. 3.19. In what follows, we explain each step.

Typing follows a similar identification method as for pencil detection, except for the need for an extra step needed to eliminate 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 threshold image the hull is applied to in
Chapter 3. Methodology

order to close the holes within the table.

Figure 3.15: Images used as input for masking the background of the keyboard image

(a) Binary image for keyboard color  (b) Binary Image of table with convex hull applied.

Figure 3.16: Figures (a) and (b) display the two components required to determine if typing may be present.

(a) Keyboard binary image after masking.  (b) Skin regions after color extraction.

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
Chapter 3. Methodology

and fit the cropped grayscale image to the KNN model trained to classify keyboards, monitors and mice.

Figure 3.17: 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.
Figure 3.18: A flowchart representing the steps to extracting the keyboard and discovery of potential typing.
Chapter 3. Methodology

1 if Looking for "Keyboard, Table" then
2 table ← Calculate Convex Hull of Table Binary Image
3 contours ← Bitwise And Table and Keyboard Binary Images
4 for All Contours do
5     Check Requirements
6     if Requirements are Met then
7         centroids ← Search for nearby Centroid
8         if No Centroid Nearby then
9             centroids ← Append Centroid
10     end
11     keyboard ← Fit ROI to KNN
12     if Keyboard is Found then
13         if Hands over Keyboard then
14             if Motion is Found then
15                 hist_list ← Calculate and Append Histograms
16                 frame ← Draw Rectangles
17             end
18         end
19     end
20 end
21 return frame, hist_list, centroids
22 end

Figure 3.19: Definition for function Context() part 2, typing.

3.3.4 Talking

We provide a flowchart for detecting the talking activity in Fig. 3.22 and provide pseudocode in Fig. 3.23. In what follows, we summarize each step.

To identify talking, the steps are slightly different. The skin is found using the same technique as the pencil, first converting the frame to YCrCb space and extract the range which preserves only areas of color matching the defined minimum
Chapter 3. Methodology

and maximum. 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. Code is provided in appendix for the person object.

Figure 3.20: Correctly identified faces (left) and skin regions that were sent to KNN (right).

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 then passed to the 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. The golden ratio is commonly used in art and design applications, and is not often used in engineering practice. Application of this ratio adequately captures the mouth at any angle we are looking for. This area is checked for motion, and as the other methods, the motion vectors are calculated into a 1d histogram.
Chapter 3. Methodology

Figure 3.21: Process of selecting a region for KNN and the output.
Figure 3.22: A flowchart representing the steps to identifying a face.

```
if Looking for Talking then
  person_object ← PersonDetect()
  for All Contours do
    Check Requirements
    if Requirements are Met then
      centroids ← Search for Nearby Centroid
      if No Centroid Nearby then
        centroids ← Append Centroid
      end
      face ← Fit ROI with KNN
      if face is found then
        if motion found in \( \frac{\text{face_height}}{1.618} \) then
          hist_list ← Calculate and Append Histograms
          frame ← Draw Rectangles
        end
      end
    end
  end
return frame, hist_list, centroids
```

Figure 3.23: Pseudocode for function Context() part 3, talking.
Chapter 3. Methodology

3.3.5 Feature vectors

The feature vectors are extracted every 3 frames as given in Table 3.1. 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. Here, we note that we have a fixed size feature vector for each object. If a feature vector is missing, then the assumption is that the corresponding object is missing.

3.4 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.

<table>
<thead>
<tr>
<th></th>
<th>Writing</th>
<th>Typing</th>
<th>Talking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>Table A.4, Paper A.1, Pencil A.2</td>
<td>Table A.4, Skin A.5, Keyboard A.3</td>
<td>N/A</td>
</tr>
<tr>
<td>Object Check</td>
<td>40&lt;area&lt;2000 and aspect ratio &gt;1.2 or &lt;0.5</td>
<td>200&lt;area&lt;8000 and Has 3+ corners</td>
<td>area&gt;1000 and 0.5≥aspect ratio≥1.5</td>
</tr>
<tr>
<td>Context Check 1</td>
<td>Pencil within table region</td>
<td>Keyboard detected on table</td>
<td>Face is detected</td>
</tr>
<tr>
<td>Context Check 2</td>
<td>Pencil near a piece of paper</td>
<td>Hands inside keyboard bounding box</td>
<td>Bottom part of face has motion</td>
</tr>
<tr>
<td>KNN Used</td>
<td>Yes</td>
<td>Yes</td>
<td>Color Only</td>
</tr>
</tbody>
</table>

Table 3.1: Context Conditions for Object Recognition
Chapter 4

Results

4.1 Dataset

We summarize the dataset types in Table 4.1 and the size of the dataset in Table 4.2. For model selection and proper reporting of the results, we used nested cross-validation. We report our final results using tenfold cross validation. Within the training set, we perform five-fold randomized cross-validation for model selection. For KNN, we investigated $K = 3$ to $K = 19$, the use of both Euclidean and city-block distance for determining the nearest neighbor, and the use of feature scaling.

<table>
<thead>
<tr>
<th>Writing</th>
<th>Typing</th>
<th>Talking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cropped Training Video</td>
<td>Positives Only</td>
<td>None</td>
</tr>
<tr>
<td>Raw Training Video</td>
<td>Negatives Only</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4.1: Dataset Video Types
Chapter 4. Results

<table>
<thead>
<tr>
<th></th>
<th>Talking</th>
<th>Typing</th>
<th>Writing</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Features</td>
<td>1755</td>
<td>1050</td>
<td>620</td>
</tr>
<tr>
<td>No. of Videos</td>
<td>14</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>FPS</td>
<td>60</td>
<td>26-60</td>
<td>24-60</td>
</tr>
<tr>
<td>Video Duration</td>
<td>5-16</td>
<td>5-24</td>
<td>1-39</td>
</tr>
</tbody>
</table>

Table 4.2: Dataset information for the final training dataset.

4.2 Classification Model Results

4.2.1 Results for Writing

We present two accurate pencil identifications in Fig 4.1, showing that our color + context model is working correctly in these training cases.

![Testing](image1)
![Training](image2)

(a) Testing. (b) Training.

Figure 4.1: Subfigures and display correct classification results for the color and context model of finding a pencil.

We present an example of the KNN correctly classifying writing within a frame in Fig 4.2. Additionally we present an example of not-writing correctly identified in Fig 4.4 in unseen video from the AOLME dataset. These images are cropped or censored for privacy of the study participants.
Chapter 4. Results

Figure 4.2: A correctly classified writing activity.

For writing we achieved a highest accuracy using the DNN model with an approximate percentage of 92.5% accuracy. The dataset results are above in Table 4.2, and the remaining results for each classifier are found in Tables 4.3 thru 4.6.

Figure 4.3: Results in an outside unseen video from the AOLME database.
Chapter 4. Results

Figure 4.4: Results from model in an outside unseen video from the AOLME dataset.

Pencil not writing is correctly classified as not writing in Fig 4.4 with noise correctly classified as not writing; however, misclassified as being a pencil.

4.2.2 Results for Typing

Figure 4.5: Correct keyboard identification.

In Fig 4.5 we see correct identification of a keyboard using the color + KNN + context model.
Chapter 4. Results

Figure 4.6: Correct classification in outside unseen video from AOLME dataset.

Figure 4.7: Correct classification of typing activity.

Figs 4.6 and 4.7 show the correct classification of typing activity with our model, with Fig 4.6 being taken from unseen video from the AOLME dataset, and Fig 4.7 coming from our validation dataset.
Chapter 4. Results

Figure 4.8: Correct classification of not typing in difficult case.

Figure 4.9: Correct classification of typing activity in difficult case.

The figures above display correct classification of no typing where movement is present near a keyboard in Fig 4.8, displaying correct results from our KNN model, and Fig 4.9 displays the correct classification of typing inside a difficult case where the monitor is partially occluding the keyboard.
Chapter 4. Results

4.2.3 Results for Talking

Figure 4.10: Subfigures and display correct classification results for the color and correct KNN results for the face.

In Fig 4.10 we see the two classification precursor steps for identifying talking successfully retrieved. First is the skin region necessary for finding the face in the KNN 4.10a, secondly we see the output of the KNN model during training after the face has been correctly identified in 4.10b.

Figure 4.11: Correct classification of face KNN during validation.
Chapter 4. Results

Fig 4.11 displays correct face locations versus other skin areas being identified during the validation phase.

Figure 4.12: Subfigures and display correct classification results for no talking and talking.
Chapter 4. Results

We present results of our validation set where talking has been successfully identified in three difficult cases. These cases are considered difficult due to the chances of laughing or eating being confused with talking.

Figure 4.13: Correct classification of talking activity in difficult case in unseen video.

Finally, we present a difficult classification in unseen raw video from the AOLME dataset with accurate classification in Fig 4.13.

4.3 Failed Classification Examples

We present several failures of our models. First, we see a failure in the keyboard classification due to the color of the ledge of the table still being classified as a keyboard after passing through the KNN. The appearance of skin also confuses the model.
Figure 4.14: Incorrect classification of table and legs as keyboard.

(a) Face KNN Failure.  
(b) Face KNN miss.

Figure 4.15: Subfigures and display incorrect functionality of the face KNN from skin and blank space.
Chapter 4. Results

Figure 4.16: Classification of a mouse as a keyboard.

We present three more KNN failures in object detection in Fig 4.15 and Fig 4.16, where objects have been misclassified using their respective KNN models. The features of these models are reviewed with their respective accuracies in Table 4.3. Keyboard classification accuracy was roughly 84% and faces were approximately 90% accurate in classification, with most confusing coming from solid areas of skin in the case of faces, and mice being confused for keyboards in the keyboard model.

Figure 4.17: Table masking issue.
Chapter 4. Results

We present a total failure of object recognition in the two Figures 4.17 and 4.18, where the masking of the table has been confused by the whiteboard, and the classifier was passed information that was not actually on a table. This confused the classifier, which has reported everything as a keyboard.

Additionally, we see another failure in training our KNN where the keyboard is seen to be a mouse in Fig 4.19. As can be seen in the results Table 4.3 containing the keyboard confusion matrix, this makes sense as most of our mice were classified
as being keyboards, indicating we need a better training set for this data.

In Fig 4.20, we observe a classification that could not be made with the KNN model due to a hand being on the chin of the girl on the right. This kind of failure presents repeatedly in the research, especially in the unseen data. Having hands too close to faces, or someone else’s arm too close to a face will cause a classification error as it confuses the KNN model, and additionally violates our area requirement stated in the previous section.
Chapter 4. Results

(a) Writing Failure.  (b) Typing Failure

(c) Talking Failure.

Figure 4.21: Subfigures displaying incorrect classifications from all models.
Chapter 4. Results

We present failures of classification from all three models, which failed in the KNN and DNN methods. These were not difficult cases but likely had their motion confused due to a poor training set or otherwise.

<table>
<thead>
<tr>
<th>Faces</th>
<th>Keyboards</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of Positives</td>
<td>1209</td>
</tr>
<tr>
<td>No. of Negatives</td>
<td>399</td>
</tr>
<tr>
<td>Accuracy</td>
<td>89.75%</td>
</tr>
</tbody>
</table>

Table 4.3: KNN Models for Typing and Talking

<table>
<thead>
<tr>
<th></th>
<th>Face</th>
<th>No Face</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face</td>
<td>233</td>
<td>15</td>
</tr>
<tr>
<td>No Face</td>
<td>18</td>
<td>56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Key</th>
<th>Mon</th>
<th>Mou</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>18</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Mon</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Mou</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.22: Confusion matrices for object KNN. Mon stands for Monitor.

<table>
<thead>
<tr>
<th>Talking</th>
<th>Typing</th>
<th>Writing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>84.62%</td>
<td>70.96%</td>
</tr>
<tr>
<td>K for KNN</td>
<td>11</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 4.4: KNN Results

<table>
<thead>
<tr>
<th>Talking</th>
<th>Typing</th>
<th>Writing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>52.99%</td>
<td>63.38%</td>
</tr>
<tr>
<td>K for KNN</td>
<td>11</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 4.5: Tensor-flow KNN Results
Chapter 4. Results

<table>
<thead>
<tr>
<th></th>
<th>Talking</th>
<th>Typing</th>
<th>Writing</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>655</td>
<td>108</td>
<td>153</td>
</tr>
<tr>
<td>F</td>
<td>49</td>
<td>162</td>
<td>74</td>
</tr>
</tbody>
</table>

Figure 4.23: Confusion matrices for KNN Results.

<table>
<thead>
<tr>
<th></th>
<th>Talking</th>
<th>Typing</th>
<th>Writing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>99.72%</td>
<td>82.52%</td>
<td>92.47%</td>
</tr>
<tr>
<td>Batch Size</td>
<td>100</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>Activation Model</td>
<td>Relu</td>
<td>Selu</td>
<td>Selu</td>
</tr>
<tr>
<td>Neurons</td>
<td>100</td>
<td>50</td>
<td>70</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.01</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Hidden Layers</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Regularization</td>
<td>L1/L2 Max_Norm</td>
<td>L1/L2 Max_Norm</td>
<td>L1/L2 Max_Norm</td>
</tr>
</tbody>
</table>

Table 4.6: DNN Results

4.4 Discussion

The classification results were found to be surprisingly accurate in some models and not as accurate in others. In the case of writing, we decided to stop and not further attempt to optimize the results due in part to a lack of time, but also because we wished to have a comparison to the models which included the KNN in addition to color to identify the other objects of interest. This gave us a great point of comparison to determine if the color of an object is enough to correctly identify a human activity.

In general, we found that color alone will work well under certain, inordinately specific circumstances, which include videos that are designed not unlike those in the most commonly used datasets. In these cases, we realize accurate classification because there is not much color noise in the background which could be misclassified.
as a pencil. The tabulated activity itself is found to be highly variable upon which videos are used to train the motion vectors. As many people have many different writing styles, some writing cannot be accurately classified using the motion vectors because the pencil will move so little. We see examples above of successful, good, and failure in the pencil classifier. The DNN model achieved the greatest results in terms of accuracy; however, often writing was still missed in certain cases, this is preferable to non-writing being inappropriately classified as writing. This activity detection model performed more poorly than the other two models.

Typing identification was moderately successful with most issues found in the classification of the keyboard itself, not unlike the major issues with the pencil model. Since the keyboard appears as a hole in the table, we had to take the convex hull of the table shape in an attempt to cover it to create context for the keyboard on the table. This did not always work due to the position of the monitor. If the monitor occluded any part of the keyboard, we were not able to ascertain whether a keyboard was present or not. Additionally, no classification takes place in the situation where the hull does not cover the entirety of the table region, which occurs when the monitor blocks both edges of the table. We found often our keyboard was misclassified as a mouse, and this caused several misclassifications during the training phase. Again, this was highly variable upon the dataset used for training due to different typing styles those who use the whole hand to type versus hunt and peck and whether the mouse was placed too closely to the keyboard. This model was found to be more accurate than the writing model; so long as the keyboard is identified, the typing is correctly labelled. Again, the DNN outperforms both KNN models.

For talking, we had the fewest errors. The faces were far easier to identify than the keyboard and thus finding the motion of the bottom half of the face was not difficult. Due to this, we get an incredibly high accuracy on all our models. There are some specific problems when the face is not correctly identified, usually as an
Chapter 4. Results

arm or a hand, which are most often classified as not talking, so not much change is visually observed when reviewing playback unless the debug labels are applied. This model performed best when applied to unseen datasets that were not used in training or testing. The DNN performed exceedingly well in this model, though the KNN accuracies are also quite close.

The videos used in training and testing varied between experiments, the majority of training being done on a collection of videos specifically recorded for the purposes of the ECE 633 Advanced Image Processing class at UNM. Some video from the AOLME dataset were also used in training the classifiers, although we attempted to reserve as many of these as possible for the purpose of testing unseen video data.

For the writing videos, a cropped dataset were used where each video was cut so the pencil would be clearly seen in each frame containing writing and raw video was used for training no writing. This was done so the writing videos would only record the motions of the pencil and the training set would contain no noise.

For the typing dataset, the full uncut videos were used for training of typing and no typing, though were specifically selected so that the keyboard was not obscured by the monitor. This was not realistic in the real-world video thus caused a lot of classification error when applied to the raw dataset.

For talking, closeups of people talking were used to record motion vectors of the mouth moving, again uncut video was used to record faces that were not speaking, but moving around.

When each model is applied to raw datasets we achieved some amount of success, the greatest of which being seen with talking versus not talking. Since we obtained extremely high accuracy rates with the talking vs. not talking, this was a somewhat expected result. It is believed this is due to how people tend to move their heads when speaking, and hold mostly still while doing anything else, except in the case
Chapter 4. Results

of reading. The other videos seemed to have a variable amount of error with each model, most of which lasted only a few frames before self-correcting.

Finally, the datasets used for testing and training came from different video databases, and were capable of correctly identifying the specified motions in different settings with different actions without editing. This also was somewhat successful when used on videos obtained from imagenet for writing, which means the model could potentially generalize with some amount of optimization.
Chapter 5

Conclusions and Future Work

5.1 Conclusions

We conclude that by combining object context with motion vector training, we can classify human activity within a set of unrelated raw videos with some degree of accuracy. The method can locate an actions specific location in a busy video in which many other activities are taking place. There were a few obstacles to increasing the accuracy in our model which may easily be overcome with an applied amount of optimization and a far larger and more diverse video training set.

The largest errors to overcome in our model concern appropriate identification of objects we wish to apply context to before extracting the motion vectors. These include errors present when objects are the same color or tone as the primary object we are attempting to locate in a frame. When there is another object that is similar color resembling one of the interaction objects it confuses the classifier, and simple movements like a head passing by will cause inaccurate identification of writing. We attempted to correct this with the keyboard and face by using an additional KNN to verify we are adding context to the correct object. The extra context does improve
accuracy, but even with the extra processing it still is not perfect.

The monitor created several problems for us in classification videos as it tended to block line of sight to many activities we were seeking. This, unfortunately, does not have a solution as we cannot see through objects. Most of the problems were seen with typing, since the monitor was often in the way of the keyboard and they are the same color, so they often blended together when the colors were extracted for thresholding, therefore we could not accurately locate the keyboard in the frame in all cases.

Though we obtained a >99.6% classification accuracy using a DNN trained for talking, we do not trust this number as it seems far too high accuracy of a classification rate. We believe this deserves extreme scrutiny in any future study, perhaps removal of dense vectors which are associated with head movement. The classification accuracy averages in the mid 70% for writing and typing activity using KNN seem far more realistic, and when played back on a raw dataset unrelated to our training videos, seems to match the amount of errors per frame we see during playback. We chose to use the average accuracy in these cases rather than the maximum accuracies obtained since they were not reproducible after adding to the training set.

Overall, the method is a promising step toward accurate classification of human activity as it eliminates much of the confusion of action that is seen in current state-of-the-art models by giving a location and context to the objects in a frame. The generality of the model allows it to be used on more than one dataset, and with some additional work could potentially serve as the first step in automatic classification procedures in the future.

Furthermore, we conclude that KNN is not the best classification method for this type of dataset. Due to the nature of the motions being quite minute in magnitude, many times motion vectors for a minute action are confused with noise or other
Chapter 5. Conclusions and Future Work

movements, which lead to misclassification of illumination noise as movement, an unfortunate side effect of the way dense optical flow operates.

5.2 Future Work

Given more time, we would like to apply a RBF neural network to classify the found object motion, eliminating the need for cropping video and labelling the motion vectors. Since RBFs can come up with a classification without this data, it would facilitate a more robust model. As KNN model performed poorly, a better model for classification may be a SVM or a Clustering based algorithm with some amount of dimensionality reduction, and the RBF would be an ideal replacement given those conditions.

Many issues in our method came from having too much color noise in the frame being classified as an action when it was too close to other similar objects. Though we attempted to apply a median blur to each frame to remove some illumination noise, it could not all be reduced, causing many classification errors. Lucas-Kanade should be investigated as an alternative to dense optical flow.

Another interesting addition would be to add haar cascades for object recognition as this would allow for classification of more than a single type of pencil, or single type of mouse, etc. Our model was highly limited by the fact that we only sought objects of the same color and having a library of objects to look at could greatly improve the generalization of the model in the future.

Finally, the application of an AI such as PAL at the end of the DNN classifier, we could insert a random set of objects and allow the AI to search for the objects that are found to be interacting with one another, thereby giving it a more appropriate label. For example, hands sideways pencil could be classified as drawing, or hands,
keyboard, table can be found as typing. This effectively eliminates the need for writing complex context functions and the necessitation of labelling video data, since when linked with an RBF for unsupervised learning, could automatically label the interactions and activities. This would also help with identifying noisy regions so that they could more easily be ignored in any future model.
Chapter 6

Appendices
Appendix A

Python Code for Color Detection

A.1 Paper

#Paper portion of code

def FindPaper(hsv, frame, an_object):
    # Method provides maximum customization options
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,
                                           # ideally these will be defined by a file that has these values stored,
                                           # or more ideally a file will define all the code.
                                           mask1 = cv2.inRange(hsv, (0, 0, 160), (188, 20, 240))
                                           mask1 = cv2.erode(mask1, None, iterations=2)
                                           mask1 = cv2.dilate(mask1, None, iterations=2)
                                           mask = cv2.inRange(hsv, (0, 0, 174), (180, 255, 255))
                                           mask = cv2.erode(mask, None, iterations=2)
                                           mask = cv2.dilate
                                           res1 = cv2.bitwise_and(frame, frame, mask= mask1)
Appendix A. Python Code for Color Detection

```python
res2 = cv2.bitwise_and(frame, frame, mask=mask)
res2 = res2 - res1
greyfr2 = cv2.cvtColor(res2, cv2.COLOR_BGR2GRAY)
_, thr2 = cv2.threshold(greyfr2, 210, 255, cv2.THRESH_BINARY)
thr2 = cv2.morphologyEx(thr2, cv2.MORPH_CLOSE, kernel, iterations=2)
thr2 = cv2.morphologyEx(thr2, cv2.MORPH_OPEN, kernel)
return thr2

A.2 Pencil

#Pencil portion of code

def FindPencil(hsv, frame, an_object):
    t = cv2.cvtColor(frame, cv2.COLOR_BGR2YCR_CB)
mask2 = cv2InRange(hsv, (13, 45, 60), (30, 176, 255))
mask2 = cv2.erode(mask2, None, iterations=1)
mask2 = cv2.dilate(mask2, kernel, iterations=3)
mask2 = cv2.morphologyEx(mask2, cv2.MORPH_CLOSE, kernel, iterations=4)
mask2 = cv2.GaussianBlur(mask2, (11, 11), 0)
mask3 = cv2InRange(t, (69, 125, 0), (254, 199, 105))
mask3 = cv2.erode(mask3, None, iterations=1)
mask3 = cv2.dilate(mask3, kernel, iterations=3)
mask3 = cv2.morphologyEx(mask3, cv2.MORPH_CLOSE, kernel, iterations=4)
mask3 = cv2.GaussianBlur(mask3, (11, 11), 0)
res = cv2.bitwise_and(frame, frame, mask=mask3)
```

Appendix A. Python Code for Color Detection

greyfr = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
_, thr = cv2.threshold(greyfr, 130, 240,
                      cv2.THRESH_BINARY)

th2 = PersonObject(frame)
th2 = cv2.GaussianBlur(th2, (11, 11), 0)
th2 = cv2.dilate(th2, kernel, iterations=3)
gray = cv2.cvtColor(th2, cv2.COLOR_BGR2GRAY)
_, th2 = cv2.threshold(gray, 0, 255,
                      cv2.THRESH_BINARY)

return thr

A.3 Keyboard

#Keyboard portion of code
def FindKeyboard(hsv, frame, an_object):
    min_YCrCb = np.array([0, 120, 127], np.uint8)
    max_YCrCb = np.array([144, 255, 177], np.uint8)
    # Convert image to YCrCb
    image_YCrCb = cv2.cvtColor(frame,
                                cv2.COLOR_BGR2YCR_CB)
    dark_region = cv2.inRange(image_YCrCb,
                               min_YCrCb, max_YCrCb)
    res3 = cv2.erode(dark_region, None, iterations=2)
    res3 = cv2.dilate(res3, None, iterations=2)
    res3 = cv2.bitwise_and(frame, frame, mask=res3)
    greyfr3 = cv2.cvtColor(res3, cv2.COLOR_BGR2GRAY)
    _, thr3 = cv2.threshold(greyfr3, 0, 255,
                            cv2.THRESH_BINARY)
Appendix A. Python Code for Color Detection

```python
return thr3, greyfr3
```

### A.4 Table

```
# Table portion of code

def FindTable(hsv, frame, an_object):
    min_YCrCb = np.array([181, 109, 127], np.uint8)
    max_YCrCb = np.array([255, 135, 160], np.uint8)
    # Convert image to YCrCb
    image_YCrCb = cv2.cvtColor(frame,
                                cv2.COLOR_BGR2YCR_CB)
    mask = cv2.inRange(image_YCrCb, min_YCrCb, max_YCrCb)
    res2 = cv2.bitwise_and(frame, frame, mask=mask)
    greyfr3 = cv2.cvtColor(res2, cv2.COLOR_BGR2GRAY)
    _, thr3 = cv2.threshold(greyfr3, 0, 255,
                            cv2.THRESH_BINARY)
    thr3 = cv2.morphologyEx(thr3, cv2.MORPH_CLOSE,
                            kernel, iterations = 3)
    thr3 = cv2.morphologyEx(thr3, cv2.MORPH_OPEN,
                            kernel, iterations = 2)
    return thr3
```

### A.5 Skin

```
def PersonObject(frame):
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,
                                        (11,11))
    f = np.copy(frame)
```
Appendix A. Python Code for Color Detection

\[ \text{min\_YCrCb} = \text{np\_array([75, 138, 111], np\_uint8)} \]
\[ \text{max\_YCrCb} = \text{np\_array([172, 155, 124], np\_uint8)} \]
\[ \text{min\_YCrCb1} = \text{np\_array([56, 142, 110], np\_uint8)} \]
\[ \text{image\_YCrCb} = \text{cv2\_cvtColor(f, cv2\_COLOR\_BGR2YCrCb)} \]
\[ \text{imgYCC} = \text{cv2\_GaussianBlur(image\_YCrCb, (11, 11), 0)} \]

# Find region with skin tone in YCrCb image
\[ \text{skin\_region} = \text{cv2\_inRange(imgYCC, min\_YCrCb, max\_YCrCb)} \]
\[ \text{skin\_region2} = \text{cv2\_inRange(imgYCC, min\_YCrCb1, max\_YCrCb)} \]
\[ \text{skin\_region} = \text{cv2\_bitwise\_and(skin\_region, skin\_region, mask=skin\_region2)} \]
\[ \text{skin\_region} = \text{cv2\_morphologyEx(skin\_region, cv2\_MORPH\_OPEN, kernel)} \]
\[ \text{skin\_region} = \text{cv2\_dilate(skin\_region, None, iterations=10)} \]
\[ \text{skin\_region} = \text{cv2\_morphologyEx(skin\_region, cv2\_MORPH\_CLOSE, kernel)} \]
\[ \text{skin\_region} = \text{cv2\_GaussianBlur(skin\_region, (29, 29), 0)} \]
\[ \text{skin\_region} = \text{cv2\_erode(skin\_region, kernel2, iterations=2)} \]
\[ \text{skin\_region} = \text{cv2\_morphologyEx(skin\_region, cv2\_MORPH\_OPEN, kernel, iterations=2)} \]
\[ \text{res} = \text{cv2\_bitwise\_and(f, f, mask=skin\_region)} \]
\[ \text{return res} \]
References


REFERENCES


REFERENCES


REFERENCES


[34] ______, Classification under a multivariate bernoulli: An application to pneumoconiosis, Proc. of spie medical imaging, 2006.

REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


[74] ———, *A dynamically reconfigurable dct architecture for maximum image quality subject to dynamic power and bitrate constraints*, Ieee southwest symposium on image analysis and interpretation (ssiai), 2012April, pp. 189–192.


REFERENCES


REFERENCES


REFERENCES


REFERENCES


[110] Li-Jia Li, Richard Socher, and Fei-Fei Li, Towards total scene understanding Classification, annotation and segmentation in an automatic framework., CVPR, 2009, pp. 2036–2043.
REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


[215]——, Novel algorithms for the accurate, efficient, and parallel computation of multidimensional, regional discrete fourier transforms, 10th mediterranean electrotechnical conference(melecon), 2000, pp. 530–533 vol.2.


REFERENCES


REFERENCES


REFERENCES


[248] R. Paul and M. Pattichis, *Nested random phase sequence sets: A link between am-fm demodulation and increasing operators with application to cardiac image analysis,*
REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES
