Unified framework for human-object interaction recognition

Jacob Chen\textsuperscript{a}, Haidar K. Khan\textsuperscript{b}, Ivan Bogun\textsuperscript{c} and Eraldo Ribeiro\textsuperscript{d}

\textsuperscript{a}Electrical Engineering, University of Maryland, College Park, Maryland; jchen114@terpmail.umd.edu
\textsuperscript{b}Computer Engineering, State University of New York at New Paltz, New Paltz, New York; hkaln47@hawkmail.newpaltz.edu
\textsuperscript{c}Computer Sciences, Florida Institute of Technology, Melbourne, Florida; ibogun2010@my.fit.edu
\textsuperscript{d}Computer Sciences, Florida Institute of Technology, Melbourne, Florida; eribeiro@fit.edu

ABSTRACT

We propose a framework for human-object interaction recognition in videos. Our method uses trajectories of body parts as a main source of information for classification. More specifically, we represent interactions by a combination of the trajectory’s spatial information and the presence of objects. We exploit the appearance of an object in multiple frames to improve object classification and ultimately, interaction recognition. Our experiments show that: (a) using multiple frames increases object recognition, (b) implementing a specialized dynamical time warping kernel allows for accurate classification of the trajectories without discarding spatial information, (c) the combination of motion and object cues, using multiple kernel learning, improves classification. Our algorithm achieves results comparable or superior to the state of the art.

Keywords: Interaction Recognition, Action Recognition, Human-Object Interaction, Video Analysis, Multiple Kernel Learning

1. INTRODUCTION

The real world is full of complex interactions between actors, usually humans, and a wide variety of objects. Most human-object interactions are done using the hands: drinking, opening doors, and writing are a few examples of these actions. The identification of these interactions is an important and open problem in computer vision. Machine identification of interactions would open the door to a multitude of applications. The applications that come to mind are mostly security related, such as advanced surveillance systems and automated video annotation. However, robust frameworks for interaction recognition could pave the way for highly sophisticated human-machine communications, with machines reacting to the different types of interactions taking place around them.

With the motivation to solve this problem established, we look at the complexity of the problem. As mentioned previously, interactions are highly varied, with nearly every aspect of the interaction varying from action to action. We propose looking at a subgroup of human-object interactions, those done with the hands, which we believe account for the majority of all human-object interactions. We then attempt to identify these interactions using a combination of motion and appearance information. Our method relies on the fact that the consistent aspect of the sub-group of interactions we examine is that they involve the hands and objects. The uniqueness of a class of interactions is accounted for by how the hands move and what the objects are. To extract this information, our method requires the detection of both the hands and the human in a video. It also needs information about the location of the action object in one or more frames of the video. Motion information is captured using a tracking algorithm to construct trajectories for the hands and the body. Appearance information about the action object will be extracted using a bag of words model with Probabalistic Latent Semantic Analysis. Finally, we use Multiple Kernel Learning (MKL) to combine complementary information from trajectories, velocities, and appearance.

The framework we describe here requires minimal annotation; only the location of the action object in one frame is needed. However, our goal is to move from this semi-supervised setting to a completely unsupervised framework by retrieving this information from the generated trajectories.
This paper is organized as follows; Section 2 describes some of the related works and Section 3 and Section 4 cover the extraction of appearance-based information and motion-based information, respectively. Section 5 discusses MKL and combining the three types of information, and Section 6 presents our experiments.

Figure 1: Overview of our framework. We use automatic trajectory extraction to extract spatio-temporal information about key objects from the video containing the interaction. We also extract appearance information using Bag of Words with Probabilistic Semantic Analysis. We then combine all this extracted information using Multiple Kernel Learning to cluster or classify interactions.

2. RELATED WORK

Action recognition is a well-known problem in computer vision. Human-object interaction recognition, as a special case, received sufficient attention.\textsuperscript{7,11,14} Methods mostly differ in terms of the information extracted, ranging from using low-level features like optical flow\textsuperscript{14} to more global ones like silhouettes or spatio-temporal volumes.\textsuperscript{1}

Our approach is inspired by the work of Prest, Ferrari and Schmid (2011), which uses histograms computed from relative trajectories for classification.\textsuperscript{16} They propose that similar interactions will share similar trajectories in the spatio-temporal volume and thus would be sufficient for recognition. In this work, the authors use ideas from object detection,\textsuperscript{6} and tracking\textsuperscript{18} to detect and track parts of the object of interest. After trajectories are extracted, histograms of relative position, area, and velocity are extracted and a Support Vector Machine classifier is learned.

The work by Gupta, Kembhavi, and Davis (2009) uses a probabilistic model over velocity profiles and object appearance for interaction classification.\textsuperscript{11} They explicitly model time frames where the object is being reached, grasped, and put back. This allows for accurate localization.
3. APPEARANCE INFORMATION

Before we discuss the methods used to classify objects in images, we will first discuss the necessity of having appearance based information about the object present in the image for our algorithm. Given information about the object, we are able to associate specific types of interactions with that object. Thus, obtaining information about the object is an important step in our algorithm. It provides context for the video, allowing us to distinguish between interactions whose motion information may appear similar. We can use that information in order to attain greater precision and confidence in our classification of the interaction. A reliable object classifier is necessary for an apt interaction classifier.

In order to obtain a reliable object classifier, we combined two well known methods together. These are the Bag-Of-Words (BOW) model for image classification and Probabilistic Latent Semantic Analysis (pLSA) similar to the approach presented in Sivic et al. (2005) \(^{17}\) The approach is illustrated in Figure 2. Our results proved to be comparable to their results with respect to the quality of object classification. The descriptors used to construct the corpus necessary for BOW varies depending on the context of the classification. Thus, the chosen type of descriptors is what differentiates our approach from the used by Sivic et al. (2005). \(^{17}\)

![Figure 2: Model of our BOW approach with edge descriptors](image)

3.1 Bag Of Words

The BOW model is well known in computer vision for its application in image classification. The BOW model originated as a text based processing model to classify documents based on the frequency of the occurrence of words in that document. Similarly, image features can be treated as “words” that occur in the image. As mentioned before, the method to choose those words varies depending on the context of the classification. Similar words are grouped together using clustering algorithms. The number of words to use depends on how well the video is represented by them. Usually the number of words to use is determined by the highest accuracy of the classification. Once the corpus is obtained we can then reconstruct the entire video using the vocabulary in the corpus. Each video can then be described by the frequency of the occurrence of each image feature that pertains to that video.

The motivation for using the BOW model is that we are interested in classifying entire videos with respect to the object in the video. Because the video is made up of many frames containing the object, there will be a large number of similar features. Taking advantage of this, we construct a quality corpus thereby achieving greater accuracy with regard to classification.
3.2 Probabilistic Latent Semantic Analysis

pLSA is a text based analysis that is used to find the semantic meaning of the document given the occurrences of its words. As mentioned above, we are assuming that each object should be represented by its own distinguishable frequency vector. These frequency vectors model a distribution that we can describe with a probabilistic model. Thus, we are able to use the Expectation Maximization (EM) algorithm to place each video inside of a probabilistic distribution. We can then classify the video based on the probabilities of each video belonging to one of the categories.

The EM algorithm has two iterative steps that follow the Initialization step, namely, the Expectation Step and the Maximization Step. The model we will be using is the aspect model in asymmetric parameterization shown in Figure 3.

![Figure 3: Graphical representation of the aspect model in asymmetric parameterization](image)

We will now introduce probabilities and terms that are used in pLSA:

1. $P(d_i)$ is used to denote the probability that a word occurrence will be observed in a particular document $d_i$
2. $P(w_j | z_k)$ denotes the class-conditional probability of a specific word conditioned on the unobserved class variable $z_k$
3. $P(w_j | z_k)$ denotes a document-specific probability distribution over the latent variable space
4. $n(d_i, w_j)$ denotes the number of times word $w_j$ occurs in document $d_i$.
5. $n(d_i)$ refers to the document length

The initialization step of the algorithm gives random initial probabilities $P(z_k | d_i)$, but we will modify the initial probabilities when we propagate information gained from one descriptor to the other later in this report. The following EM formulas are taken from Hofmann (2001).

The Expectation formula is given by:

$$P(z_k | d_i, w_j) = \frac{P(w_j | z_k)P(z_k | d_i)}{\sum_{l=1}^{K} P(w_j | z_l)P(z_l | d_i)} \quad (1)$$

When we implemented the Maximization formula we received numerical errors for our entries in $P(w_j | z_k)$. In order to resolve this issue, we implemented Laplace Smoothing for our M-Step equations in order to guarantee that we would not end up with such errors. Our new M-Step equations are as follows:

$$P(w_j | z_k) = \frac{\sum_{i=1}^{N} n(d_i, w_j)P(z_k | d_i, w_j) + \gamma_j}{\sum_{j=1}^{M} \gamma_j + \sum_{m=1}^{M} \sum_{i=1}^{N} n(d_i, w_m)P(z_k | d_i, w_m)} \quad (2)$$
\[ P(z_k | d_i) = \sum_{j=1}^{M} n(d_i, w_j) P(z_k | d_i, w_j) \]

In our experiments we used \( \gamma_j = 1 \quad \forall j \).

We are particularly interested in \( P(z_k | d_i) \) since those probabilities represent our confidence of a video belonging to a class. Once these probabilities are obtained, we can then determine which class each video belongs to and compare the predicted classes with the ground truth. Afterwards we are able to construct a kernel which describes the similarity between videos in terms of the objects being used. This is achieved by applying the Eq. (4). We will then use the information from this kernel to finally classify the interaction in the video. Formally, the kernel is defined by

\[ K(r_i, c_j) = \sum_{k=1}^{M} P(z_k | d_i) P(z_k | d_j). \] (4)

4. MOTION DESCRIPTORS

In a video containing an interaction between an actor and an object, much of the characteristic information about the interaction is contained in the motion of the actor, the object, and sometimes parts of each. Thus, it is essential for a system that aims at identifying interactions to extract spatio-temporal information about objects of interest. The spatio-temporal information is in the form of trajectories which record the position of an object over time. The position is given by \( P(x, y, t) \), where \((x, y)\) is the coordinate location of the center of the object, and \(t\) is the time, measured in frames, the object was at that position. Since we are dealing with the projection of a three-dimensional environment onto a two-dimensional environment using an imaging device, we lose a sizeable amount of information about the position of the object, specifically, a third coordinate axes \(z\).

We note that these trajectories are time-series of differing lengths, which presents some challenges that will be addressed later in the paper.

A simple way to extract the trajectory of an object of interest from a sequence of frames is tracking by detection. This method simply involves detecting the object in every frame of the sequence and linking the detections together to form a trajectory. This method falls short in two areas. Firstly, it is unable to deal with partial occlusions of the object in any frame of the sequence without resorting to a naive method like interpolation. Secondly, the accuracy of the trajectory is directly related to the accuracy of the detector. Any frames for which the object is not detected or where a false positive is detected will cause either interpolation error or an impossible situation where the object moves a great distance between two successive frames. Since we will be using a trajectory extraction method based on tracking by detection, it is essential to have a robust object detector.

4.1 Object Detectors

For our object detection, we trained our detectors using the approach detailed in Girshick, Felzenszwalb, and McAllester (2010). A brief summary of the approach is given here. The detectors we trained are deformable part-based models, where each detector consists of a number of parts connected in a star formation. We trained the detectors using a version of Multiple Instance Learning - Support Vector Machines called Latent Support Vector Machine (LSVM). LSVM extends the normal Soft-Margin SVM by introducing latent variables which are optimized for along with the separating hyperplane. Details of the formulation can be found in Felzenszwalb et al. (2010), but a useful intuition is that Soft-Margin SVM can be described as a special case of LSVM where the latent variables are known. The training images used for LSVM are weakly labelled, only the bounding box around the object is labelled. The location of the parts are treated as latent variables. As such, the optimal parts for an object are automatically selected and optimized. This is beneficial because it avoids a large amount of annotation which is costly and time-consuming and also avoids the problem of specifying parts which do not result in the optimal model.
4.1.1 Deformable Part-Based Models

The models we used as detectors consist of a root filter and a number of part filters. The root filter encompasses the whole object at some coarse resolution, while the part filters cover smaller parts of the object at higher resolutions. Felzenszwalb et al. (2010) propose that applying part filters at higher resolutions results in higher accuracy because it allows the part filters to capture finer features. Essentially, the root filters capture the general properties of the object, while the part filters focus on specific parts and details.

The detector uses a sliding window approach acting as a linear filter applied to all positions and a number of scales of an image. The root filter is applied to the feature map of the image at its native scale, and the response to the filter is generated. Each part filter is then applied to the feature map of the image at twice the resolution. The resulting responses are combined with the response from the root filter to predict the possible locations for the object in the image. Each potential location for the object is scored by the magnitude of the response from the filter and penalized by a “deformation cost” that is determined based on the position of the part with respect to the root.

Figure 4(b) shows, starting from the left, the root filter, each part filter, and the deformation of parts from the root. The features for the model are visual representation of Histogram of Oriented Gradients (HOG) features. Figure 4(c) shows the hypothesis for the location of the object and its parts based on the responses to the filters.

4.1.2 Histogram of Oriented Gradients

Our detectors used HOG based on Dalal and Triggs (2005) to extract feature vectors from images. A HOG feature map is constructed by extracting the magnitude and orientation of the gradient of each pixel in the image. To reduce the size of the features, the image is divided into a grid of square cells and the gradient of each cell is determined by the pixels in the vicinity and inside the cell.

Let \( k > 0 \) be the length of the side of a cell in a \( w \times h \) image. Let \( C(i,j) \) define a cell in the grid of cells, where \( 0 \leq i \leq \lfloor \frac{w-1}{k} \rfloor \) and \( 0 \leq j \leq \lfloor \frac{h-1}{k} \rfloor \). Let each pixel \((x, y)\) have \( \theta(x, y) \), the orientation of the intensity gradient, and \( r(x, y) \), the magnitude of the intensity gradient. The gradient orientation is discretized into one of \( p \) values using one of two methods.

\[
B_1(x, y) = \text{round} \left( \frac{p\theta(x, y)}{2\pi} \right) \mod p \tag{5}
\]

\[
B_2(x, y) = \text{round} \left( \frac{p\theta(x, y)}{\pi} \right) \mod p \tag{6}
\]

Then, the magnitudes are used to define the feature map in the following manner:

\[
F(x, y)_b = \begin{cases} 
   r(x, y) & \text{if } b = B(x, y) \\
   0 & \text{o.w.} 
\end{cases} \tag{7}
\]

Each pixel then contributes to the orientation and magnitude of the gradients of the four cells around it. The HOG method for feature extraction is a popular one, achieving good results and also straightforward in implementation and interpretation.

4.2 Trajectory Extraction

In this work, we propose using the trajectories of objects of interest to classify interactions between actors and objects. Our trajectories will describe the motion of parts, such as the hand, in relation to other parts, such as the body. The construction of these trajectories will involve a combination of tracking by detection and optical flow. Essentially, Optical flow gives an estimation of the movement of pixels from one frame to the next. Using only detections to form trajectories is problematic because the detections of false positives cause the linking of detections over time to become difficult and inaccurate, especially when detections are sparse. We will augment our detections with optical flow to give the tracking algorithm greater accuracy in following the object of interest.
Figure 4: (a) Original image from the INRIA Person Dataset. (b) Root filter, part filters, and deformation costs for each part. Models generated from. (c) Each filter is applied to the image at every point at different scales. The root filter is applied at the image’s native scale, while the part filters are applied at twice the resolution. Locations for greatest response to the filters are marked by red boxes for the root filter and blue boxes for each part filter. (d) A final prediction for the location of the object is made based on the locations of the parts and the root.
Algorithm 1 Tracking Algorithm

**Input:** Sequence of frames $f_1...f_n$, detections $D^f_i$ for $i \in 1...n$

**Output:** Set of tracks $T$

1. Initialize a new track $T$ for each detection $d \in D^f_i$, where $f$ is the first frame with a detection, and add it to $T$
2. **For Each** frame $f_i$ for $i \in 2...n$
3. **For Each** track $T \in T$
   
   \[ T_p(f) := \text{position of } T \text{ in frame } f \]

4. Calculate the optical flow, $OF$, between $f_i$ and $f_i$, localized to $T(f_{i-1})$
   
   \[ T(f_i) := T(f_{i-1}) + \text{mean}(OF) \]

5. If a detection $d \in D^f_i$ overlaps with $T(f_i)$, remove $d$ from $D^f_i$
   
   assign $T_p(f_i)$ a score indicating how well it is supported by the detection.
6. **End For Each**
7. **For Each** remaining detection $d \in D^f_i$
8. Create a new track $T$ at $d$ and add it to $T$
9. **End For Each**
10. **End For Each**

4.2.1 Tracking Algorithm

Our tracking algorithm is based strongly on the algorithm proposed by Prest et al. (2011). The algorithm takes detections at any point in a video and propagates them forwards and backwards using estimations based on optical flow. In the worst case, this method only requires one detection in one frame of the video to create a trajectory. This is especially beneficial for cases where detections are sparse, which can occur when trying to detect small objects or objects that have high degrees of deformation, such as the hand.

The tracking algorithm is shown in Algorithm 1. The algorithm begins by creating a set of tracks based on the detections in the first frame where detections exist. Then, for each initialized track, two major steps are executed. The first step is finding the next position for a track by using optical flow to estimate the movement of pixels in a window around the last location of the track. Then the detections in that frame are compared with the estimated positions of the tracks. Any overlapping detection and estimation causes the score of the track to increase. If the estimated position of a track does not overlap with any detections, the score of the track is penalized. This overlapping also removes that detection from the set of detections for that frame. In the second step, the rest of the detections which did not overlap with any estimated positions for tracks begin their own tracks starting at that frame. These steps are repeated for every frame of the video. The whole process is repeated twice for a sequence of frames, once forward and once backward. The tracks generated from the forward pass are matched with the reversed tracks generated by the backward pass and concatenated.

4.2.2 Optical Flow

Optical Flow provides a measurement of the relative motion between an observer and the scene. To calculate optical flow, we used the code associated with the work by Brox and Malik (2011). Since calculating optical flow is a computationally expensive operation, we will localize the calculation to a sub-window around detections and positions in the trajectories. This will allow us to calculate the mean displacement of a small area of pixels which represent the object of interest. Figure 5 shows the displacement vectors in a 50 $\times$ 50 pixel window between two frames.

4.2.3 Trajectory Refinement

Our tracking algorithm produces many trajectories, including trajectories that are initialized on false detections. In order to deal with this issue, we had to introduce methods to remove these extraneous trajectories. A simple way to eliminate the trajectories initialized on static objects in the background is to filter the trajectories based on the distance travelled. Let $T$ be a trajectory defined as a vector of positions $[p_1, p_2, p_3...p_n]$. We calculate the distance travelled by $T$ as $d_T = \sum_{i=1}^{n-1} |p_{i+1} - p_i|$. We defined a threshold value $\lambda$ to differentiate
between static trajectories and dynamic trajectories. A trajectory $T$ is defined as a static trajectory if $d_T \leq \lambda$ and dynamic otherwise.

### 4.2.4 Trajectory Normalization

Until this point, trajectories have been defined as vectors of absolute pixel positions in an image. However, this is unsuitable for our final goal of classifying actions based on trajectories because there is no way to compare two trajectories that are defined as absolute pixel positions. Instead, we will redefine our trajectories to some point common to all the frames in the video. For a trajectory $T_1$ defined at positions over the time period $0 \leq t_1 \leq n$, and $T_2$ defined over $0 \leq t_2 \leq n$. We can redefine $T_2$ with respect to $T_1$ using the following method:

$$T_2'(i) = T_2(i) - T_1(i) \text{ for } i = 0, ..., n$$ (8)

Shifting the reference point from the top-left corner of the image to the position of $T_1$ at every frame effectively allows our method to be invariant to translation of the imaging device. We also need to achieve invariance to scale in order to compare two trajectories of which one is generated from a video where the object of interest is closer to the imaging device. The method we use to achieve this is also intuitive. First, we find $d^*$, the maximal Euclidean distance from the set of positions in our shifted trajectories $T_2'$. We then divide every position vector in $T_2'$ by $d^*$ element-wise to generate a new normalized and scale-invariant trajectory $T_2''$. Essentially, the distance of every position $[p_1, p_2, p_3, ..., p_n]$ in $T_2''$ from the new reference point falls within $(-1, 1)$.

For our final goal of classifying actions, we theorize that a large amount of characteristic information about the action is contained in the trajectories of the hands. We will use these methods to redefine the trajectories of the hands with respect to a reference point such as the mid-section of the body.

Figure 5: Visualization of displacement vectors for pixels between two frames generated using Large Displacement Optical Flow localized to a detection window.

### 4.3 Action Classification

The final goal of this work is to classify actions based on a mixture of appearance and motion information. However, we first tested classification based on motion and appearance individually in order to fully understand
the effects combining the two types of information has on accuracy. We also aimed to generate classification results using supervised and unsupervised learning methods. Classifying time-series of different lengths is a challenging problem, which requires the use of specialized kernel methods for comparing two time-series.

For our supervised learning classification, we trained a Support Vector Machine (SVM). We extended the classification to multiple classes using a one-vs-all approach. We also used a voting method to establish a label for a testing video based on the classification of each trajectory extracted from the video.

4.3.1 Time Series Alignment Kernel

Our trajectories are time-series of position vectors, and in order to compare two time-series they must be the same length. It has been shown that using naive methods like resampling are not sufficient for solving this problem. Instead, the use of Dynamic Time Warping (DTW) is required. DTW is a non-linear time warping of two time-series which finds the optimal path that minimizes the accumulated distance between the two time-series. The kernel method we used is based on the Dynamic Time Alignment Kernel from the work by Noma (2002). The formulation used is as follows. Let \( x = (x_1, x_2, \ldots x_n) \) and \( y = (y_1, y_2, \ldots y_m) \). Let \( A(x, y) \) denote all possible alignments, \( \pi \), between \( x \) and \( y \) where \( |\pi| = p \). \( \pi_1 \) is a vector of indices of \( x \) from 1...n in \( p \) equal-sized increments and \( \pi_2 \) is a vector of indices of \( y \) from 1...m in \( p \) equal-sized increments. Essentially, \( \pi_1 \) and \( \pi_2 \) provide a way to compare each element in \( x \) with an element in \( y \).

\[
K_1(x, y) = \arg\max_{\pi \in A(x, y)} \frac{1}{|\pi|} \sum_{i=1}^{|\pi|} \exp \left( \frac{||x_{\pi_1(i)} - y_{\pi_2(i)}||^2}{-\sigma^2} \right) 
\]

This method uses the Gaussian Kernel and selects the best alignment between \( x \) and \( y \). There are other methods which involve taking the sum the scores of all alignments, but we found this kernel to perform much better for our purposes. With this kernel, we have all the tools needed to extract trajectories, normalize and scale them, refine them, and compare them.

Another piece of motion information we extracted from the videos is the velocities of the trajectories. Since our trajectories are discrete-time signals, we used a difference method to calculate the velocities. The resulting velocities are also time-series, and the same kernel method is used to compare them. We will discuss later how we combined trajectories and velocities and used them separately to classify actions.

5. MULTIPLE KERNEL LEARNING

The trajectories, velocities, and object information used in the videos carry complementary information about the interaction being performed. Mathematically, via kernels, we can define the similarity of two videos in terms of the aforementioned characteristics. Using ideas from multiple kernel learning we are able to represent the final kernel as a linear combination of our “weak” ones and train a SVM to perform classification.

5.1 Kernel Normalization

In order to avoid scaling differences of the kernels, all kernels are normalized. Normalization proceeds as follows: let \( K(x_i, x_j) \) be a Mercer, then normalization is given by the following transformation:

\[
\hat{K}(x_i, x_j) = \frac{K(x_i, x_j)}{\sqrt{K(x_i, x_i)K(x_j, x_j)}}
\]

It can easily be shown that after Eq. (10) the kernel remains positive semi-definite.
5.2 MKL

Assume $K_1,\ldots,K_n$ are Mercer kernels normalized according to Eq. (10). Since we are looking for a kernel which is able to combine complementary information, we represent it as a linear combination of weak kernels:

$$K(\cdot,\cdot) = \sum_{j=1}^{n} \gamma_j K_j(\cdot,\cdot) \quad (11)$$

In order for Eq. (11) to be positive semidefinite $\{\gamma_j\}_{j=1}^{n}$ should represent a convex combination; a common choice is to choose $\gamma$ over the unit simplex in $\mathbb{R}^n$, that is from a set $\Gamma$ defined as

$$\Gamma = \{ \gamma \in \mathbb{R}^n \mid \sum_{j=1}^{n} \gamma_j = 1, \gamma_j \geq 0 \forall j = 1,\ldots,n \}.$$  \quad (12)

Different methods were proposed on how to choose $\gamma$, however the central idea is to choose $\gamma$ so that the resulting kernel is aligned with the "ideal kernel". Let $y = [y_1,\ldots,y_n]^T$, $y_i \in \{-1,1\}$ $\forall i$ be a set of labels. The ideal kernel is defined as $yy^T$. Kernel alignment is a similarity measure between two kernels, it is defined as follows: let $K_1,K_2$ be two kernels, then their alignment is given by

$$A(K_1,K_2) = \frac{\langle K_1, K_2 \rangle_F}{\sqrt{\langle K_1, K_1 \rangle_F \langle K_2, K_2 \rangle_F}} \quad (13)$$

where $\langle \cdot, \cdot \rangle_F$ is a Frobenius norm. In our experiments we found that a simple heuristic to choose weights as a normalized similarity to the ideal kernel\footnote{Equation (14)} gave the best results:

$$\gamma_j = \frac{A(K_j, yy^T)}{\sum_{i=1}^{n} A(K_i, yy^T)} \quad \forall j = 1,\ldots,n \quad (14)$$

6. EXPERIMENTS

6.1 Experimental Setting

We conducted our experiments on the dataset from the work by Gupta and Davis (2007)\footnote{Equation (15)} The dataset is a set of 54 video clips with each video containing one of six actions. The actions are drinking from a cup, pouring form a cup, answering the phone, dialing on the phone, spraying from a spraycan, and shining a flashlight. The dataset contains many different actors and four types of objects. The annotations available for this dataset are trajectories of head, hands, and body of the actors, along with the trajectory of the action object. Because of the limited nature of the dataset, we used Leave One Out Cross Validation (LOOCV) to evaluate our classification methods. Figure 15 shows examples of each of the interactions in the videos.

6.2 Experiments on Objects

This section will be focused on the experiments related to extracting appearance information about objects. We will first discuss how the descriptors were chosen. Next we will show the results of these descriptors from our runs of LOOCV. Afterwards, we will show the results of these descriptors from our runs in an unsupervised setting. Lastly, we will analyze the presented results and draw conclusions based on these results.

Before we begin, let us introduce the classes that each video will be classified under. These object classes are shown in Figure 6. Each video will be classified under one of these four categories. It is important to note that the dataset we were given only has a limited number of videos and therefore a limited amount of data. Furthermore, some objects occur much more frequently than other objects. Additionally, the extracted images of the objects were of very low resolution. We took into account these considerations when we settled on our method for choosing image features as well as testing.
6.2.1 Choosing the descriptors

A common method for extracting image features is to use Scale Invariant Feature Transform (SIFT)\textsuperscript{13} to extract descriptors of the image. SIFT is a very effective tool to use to extract image features since the descriptors that it gives are less susceptible to orientation, noise, occlusion, etc. SIFT was the method of choice for extracting image features in the work by Sivic et al.\textsuperscript{17} Naturally, we also used SIFT to extract our image features but discovered those descriptors resulted in poor accuracy in our classification. The reason for this was the low resolution of the images. Upon verifying the feature descriptors extracted from the image using SIFT, we saw that the descriptors were describing background noise that could not be used effectively in our classification model. An example of the kind of SIFT descriptors we obtained is shown in Figure 8c. Thus, we decided to simplify our approach by dividing the image into patches of a given size and using each patch as an image feature. Although rudimentary in nature, we were able to see a drastic improvement in the accuracy of our classification.

For our method, we obtained two types of descriptors, overlaying patches and edge magnitudes. Each type of descriptor involved breaking up the object image into patches of a given patch size. With these two descriptors, we were able to gain two different sets of information that we can combine in various ways to come up with our optimal method using consistency and accuracy as the deciding factors. We will now discuss how the overlaying patches were obtained followed by how the edge magnitudes were obtained and used as image features.

The reason for using overlaying patches is because more spatial information can be gained by combining two single patches such that each pair will co-occur within a local spatial neighbourhood. In order to obtain overlaying patches, we initially divided the image up into single patches of a given patch size. We then proceeded to combine these patches in the way shown in Figure 7. This approach of combining patches was repeated throughout the whole image. For our purposes, we chose a patch size of five pixels.

Each of the combined patches will then be reshaped into a column vector that will become a single data vector in a large data matrix. We then perform the k-means clustering algorithm to assign labels to each vector. Each cluster represents a “word”. Similar image features will be grouped together and assigned a specific “word”.

![Figure 7: Each highlighted patch represents the combination of the patches of the image.](image-url)
Each video can then be described using these “words” and a frequency vector can be made for each video. By combining these frequency vectors, another data matrix can be constructed, one which we can use for pLSA.

Edge magnitudes are another type of descriptor that will give us information regarding the actual shape of the object. In order to obtain these descriptors, we transformed the image into another image where the edges can be easily discernible. This is achieved by using Canny Edge Detection. An example of a transformed flashlight image is shown in Figure 8b.

![Figure 8: Images of a flashlight with various processing](image)

Once the image has been transformed, we proceed to break up the image into patches of a given patch size. For our project, we chose to use a patch size of fifteen pixels. The next steps are similar to the steps taken for the overlaying patches. Once the k-means clustering has finished, we are left with another data matrix describing the number of occurrences each patch occurred in a video which we are able to perform pLSA.

In order to verify the integrity of our method, we performed LOOCV to establish the upper bounds of our accuracy. The graphs in Figure 9 are different combinations for using these two descriptors with varying sizes of the corpus. The most optimal combination as well as the most optimal corpus size was determined by the level of accuracy and consistency displayed in the results for each combination. The combinations of descriptors are the following:

1. Edges only
2. Doublets (overlaying patches) only
3. Combined probabilities with random initialization for Edges and Doublets
4. Combined probabilities with propagation of information from Doublets to Edges
5. Combined probabilities with propagation of information from Edges to Doublets

We propagated information gained from one descriptor to the other by initializing $P(z_k | d_i)$ of the current descriptor to the $P(z_k | d_i)$ learned from the previous run of the EM algorithm for the previous descriptor.

From these results, the number of clusters that provided the highest accuracy was found to be 55. The accuracy versus the case where the number of clusters is equal to 55 is shown in Figure 10.

Overall, the accuracies were consistent throughout the runs with different number of clusters. Each graph displayed similar behaviour to each other in terms of increasing accuracy as number of clusters increases. The propagation of information from Edges to Doublets proved to be the highest overall accuracy followed closely by the propagation of information from Doublets to Edges. The propagation of information from Edges to Doublets proved to be the most consistent with respect to accuracy throughout the runs.

Once the upper bound of accuracy was established, we moved on to testing our approach in an unsupervised setting to find the lowest bound of accuracy. Similarly to testing our approach with LOOCV, the graphs in
Figure 11 display the different combinations for using the two descriptors with varying corpus sizes. The criteria for choosing the most optimal combination as well as the most optimal corpus size remains the same. The combinations of descriptors likewise remained the same.

From these results, the corpus size that provided the highest accuracy was found to be 40. The accuracy versus the case where the corpus size is equal to 40 is shown in Figure 12.

The reason that some of these graphs start at a corpus size of 30 is because they display similar behaviour before the corpus size reaches that size. We are more interested with larger corpus sizes. The Edges seem to vary as the corpus size increases while the Doublets are more consistent with respect to accuracy. For the cases where the two descriptors were combined, accuracy was seen to vary as the corpus size increases except for the combination where the probability distributions are propagated from Doublets to Edges.

6.2.2 Analysis

The accuracies in the unsupervised setting were shown to be less consistent than LOOCV testing for some cases. This is as expected when testing in an unsupervised setting. However, the accuracies displayed for the combination where probabilistic distributions were propagated from Doublets to Edges proved to be very consistent in both the supervised and unsupervised setting. Given this case’s high rate of consistency and accuracy in both settings, we chose this case to be our method of choice. The confusion matrices for this case
are shown in Figure 13a and Figure 13b. Once we determined our final approach, we swept through the number of frames used for each video and determined the accuracy for different number of frames. The results are shown in Figure 14a and Figure 14b.

6.3 Experimental Results

For our motion descriptor experiments, we ran four classification experiments on the whole dataset using LOOCV. The first two experiments were run on the annotated trajectories and velocities from the dataset. These experiments allowed us to establish an upper-bound for performance and measure the performance of our kernel methods. The other two experiments were run on the automatically generated trajectories and velocities to show that the method works on trajectories generated using our tracking algorithm. Let $x_i, x_j$ be two videos denote $t_i, t_j$, $v_i, v_j$, $o_i, o_j$ as their trajectories, velocities and objects respectively.

Also let $\sigma = [0.1, 1, 10, 100]^T$. Assume $K_1, K_2$ are alignment and object kernels; the final kernel is given by

$$K(x_i, x_j) = \sum_{k=1}^{4} \gamma_k K_1(t_i, t_j; \sigma_k) + \sum_{k=1}^{4} \gamma_{k+4} K_1(v_i, v_j; \sigma_k) + \gamma_9 K_2(o_i, o_j)$$

As shown in Table 1, we were able to achieve around 90% accuracy as our upper bound using annotated velocities. This indicates that with a superior tracking algorithm, our method can achieve results comparable to the annotated trajectories and still benefit from not having to annotate each frame in the video.

7. CONCLUSIONS

In this work we addressed the problem of recognizing human-object interactions from videos. Being able to combine object appearance information and the relative trajectory of the actor’s hand, we were able to represent and classify interactions from the data set by Gupta et al.\textsuperscript{11} The high accuracy of the method shows that
the approach is capable of distinguishing interactions even if they share similar objects (e.g. making a phone call, answering the phone) or similar trajectory (e.g. pouring the water, lighting a flashlight). It is also worth mentioning that the multiple kernel learning approach allows for automatic parameter fitting (spread of the gaussian kernels). The fact that our classification was unchanged for a range of different values of the $C$ parameter shows that kernels are able to represent the data in an easily separable space.

Although results look encouraging there is still a room for improvement. Firstly, we used manually labeled trajectories for the objects. In reality the object has to be detected first to be recognized. Secondly, we weren’t able to achieve same level of accuracy with automatically extracted trajectories of the hands. Algorithms for more accurate hand detection and tracking should be incorporated to solve this problem. Lastly, we didn’t touch the problem of the interaction localization. In the data set we used the videos were cut to contain only the interaction; on the large scale the interaction has to be localized from a long video.

<table>
<thead>
<tr>
<th></th>
<th>Annotated % Success</th>
<th>Automatically Generated % Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectories</td>
<td>75.0</td>
<td>51.85</td>
</tr>
<tr>
<td>Velocities</td>
<td>90.7</td>
<td>46.29</td>
</tr>
<tr>
<td>Full framework</td>
<td>96.2</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Leave-one out cross-validation results.
Figure 12: Accuracy vs Case where number of Clusters = 40 in unsupervised setting

Figure 13: Confusion Matrices

ACKNOWLEDGMENTS

The authors acknowledge support from National Science Foundation (NSF) grant No. 1263011. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

Figure 14: Doublets to Edges propagation of probabilistic information with varying number of frames

Figure 15: The six actions from the dataset in.


