# Final Report for Project "Localizing Objects and Actions in Videos Using Accompanying Text"

Johns Hopkins University, Center for Speech and Language Processing,

Summer Workshop 2010

J. Neumann, StreamSage/Comcast
C. Fermueller, University of Maryland
J. Kosecka, George Mason University
E. Tzoukermann, StreamSage/Comcast
R. Chaudhry, Johns Hopkins University
F.Ferraro, University of Rochester
H. He, Honkong Polytechnic University
Y. Li, University of Maryland
I. Perera, University of Pennsylvania
B. Sapp, University of Pennsylvania
G. Singh, George Mason University
C.L. Teo, University of Maryland
X. Yi, University of Maryland

November 4, 2010

# Chapter 1

# Summary

## 1.1 Motivation

This main focus of this report is the problem of semantic annotation of actions and activities from video using unstructured textual descriptions. The problem at hand is similar in the nature to commonly studied problems of semantic parsing of static images or general object detection and recognition. The main differences are in the role of temporal aspect of visual representations, the availability of textual descriptions and the type of contextual relationships which we plan to exploit.

Larger and larger amounts of video content are being generated everyday in the world, and the volume of video data will only increase further with the proliferation of mobile video recorders such as cell phones or flip recorders that we can use wherever we are. Given the staggering volume of data, e.g. according to the official statistics on the YouTube website, more than 24 hours of new videos are being uploaded onto their web site every minute, it becomes essential to index and describe these videos, so that consumers can easily browse and search the information they are interested in. Parallels can be seen to the development of the world wide web, where without search engines such as Google, Bing, or Yahoo it becomes impossible to find web pages that one is interested in.

In our daily life we use language not just to describe how something looks in the world or what the static properties of an object are, but mainly to communicate what someone does and how they do it. Although images can transmit a lot of information about an activity that we can interpret using our contextual knowledge about the world, the full richness of an activity can only be communicated via moving images. Automatic action recognition is very important for video-indexing and search, but also has applications for content-based browsing, as well as robotics.

Currently, when one is browsing through videos, the usual browsing mode is time-based which does not take the natural semantic organization of video material into chapters, scenes, and events into account. For example when recording

an American Football match a natural semantic organization would be the individual plays of the game, which then in turn could be organized according to their level of interest, e.g. in general touchdowns are more interesting compared to incomplete passes. Similarly, the interesting events in a soccer match are the goals, so by correctly indexing such a game, the viewer could directly navigate between the goals and other similar interesting events, creating a much more meaningful interaction with the video for the viewer.

Automatic action recognition also has implications for robotics where one would like to enable a robot to autonomously understand a scene and the actions therein. This would enable the robot to both better understand what is happening in a scene, and choose more appropriate actions plans, and also possible enable the robot to learn by observing an action. The robot would break down an action into its individual parts and then attempt to emulate the action by repeating these steps.

The type of visual and textual data available in these domains differs, but the sources of data share many commonalities. The commercial videos of arts and craft shows typically come along with transcripts and/or textual descriptions of tasks to be accomplished, are partitioned into shots and have clear temporal segmentation boundaries. Objects do not have large scale variation and activities are typically observed from limited number of viewpoints; where all these factors are typically determined by the film maker. The video data used in the human-robot interaction setting on the other hand, the temporal segmentation boundaries between individual actions are not clearly determined, objects and activities are either observed from a single viewpoint or multiple viewpoints are simultaneously available [63, 1].

What are the type of actions that we are interested in? Most of the current research in computer vision has focused on actions that can be described by an observable motion pattern alone without reference to the environment, examples for these types of actions are global motions such as walking, running, jumping, waving, clapping, boxing, etc. or simple interactions between two people and the environment, e.g, hugging, kissing, opening a door, getting up from a chair.

Unfortunately, there are many actions that cannot be uniquely described only based on their visual appearance. If we take for example, the following set of pictures depicting actions that consist of the same turning motion of the hand, but have different semantic meaning based on the environment and type of objects that the action is applied to. For these type of actions it is necessary to not just accurately describe the visual motion patterns that are present, but also the identity and appearance of the objects involved, and their spatio-temporal relationships (e.g. relative positioning, contact points over time), and the overall context where the action is occurring (e.g. indoors, outdoors). Without further external domain constraints it is very difficult to accurately determine all the extraneous variables because although all these constraints taken together nicely define the context for a given action, the large number of possible variable assignments and the ambiguity in their estimation makes it practically impossible to solve the action recognition problem in a scalable and practical manner without additional external constraints on the context.

## 1.2 Related work

The general problem of video annotation and understanding is very complex as it often involves temporal interactions between semantic entities (people, people-objects, cars etc) taking place in different and changing environments/scenes and domains. The domains and scenes, where the activities take place often provide useful contextual information regarding the type of activities or geometric layout in the scene. In the absence of textual descriptions and with availability of some labeled data, the problem of action recognition has been studied in several instances previously. In the past most of the focus of computer vision community was on the study of human actions that were characterized by movement and change of posture change, such as walking, running, jumping etc. For these types of actions the understanding of the role of features derived from motion and the development of global or semi-global motion features/descriptors was followed by different strategies for solving the supervised classification problem [18]. Recognition of activities determined by human pose have been studied in [73]. This approach is applicable only to a small number of actions and the labeling and segmentation is often painstaking endeavor. With the advances on textual processing and detection, recognition and localization of faces and people several works focused on using sources of data readily available "in the wild" to tackle the problem. Static images and captions were analyzed in [8, 52] and the problems of action recognition aligning screen plays and videos have been studied in [20], simple human movement actions were learned and recognized in movies in [49]. Powerful cues provided by presence of humans (determined by the reliable detectors) and their pose it has been demonstrated that several actions representations can be attained in static images [38] from names and verbs extracted from captions. Additional approaches demonstrating the strategy of using the unstructured textual information and images have been mostly applied in the static setting [82]. The problems of action recognition in the robotics domain has been studies typically in isolated setting mostly focusing on development of invariant motion representations using 3D motion capture systems [37] and object-action interactions in case of manipulation actions [96]

## 1.3 Our approach

In this project we specifically focus on the domain of commercial videos of arts and crafts, where we seek to develop techniques for automated annotation and labeling of the video data. The commercial videos of arts and craft shows typically have available transcripts and/or textual descriptions of tasks to be accomplished, are partitioned into shots and have clear temporal segmentation boundaries. Objects do not have large scale variation and activities are typically observed from limited number of viewpoints; where all these factors are typically determined by the film maker. In the art and craft domain we plan to use commercially available high definition videos of art and craft shows and their associated transcripts. For the activities of daily living we intend to start

working with cooking activities as available in the University of Rochester - Activities of Daily Living and CMU Kitchen data sets ([63, 1]). From the perspective of computer vision tasks, we will assume that we have a set of videos and the associated set of verbs and nouns extracted from the transcript. These descriptions will be extracted by linguistic component of the project described in Part I. The ambiguities which we will need to deal with are related to the fact that not all verbs and nouns extracted have visual counterparts and some verbs and nouns may be missing or only their synonyms appear. This is similar to the ambiguities encountered in names and faces associations studied in [8].

In the domain of learning and recognition of manipulations actions there is a close interaction between the type of motion hands or human body undergo, shape of the hand and tool being held as well as object being manipulated. We study and model these interactions and cues explicitly and exploit them in the action recognition framework. In the first stage we use the previously developed object detectors [29], hand detectors and motion descriptors [49] to gather some evidence about presence of individual cues in the key frames and also propagate this evidence via tracking and detection across the sequence. We then model the interactions in the Conditional Random Field [44] framework and formulate the video annotation as a problem of most likely sequence assignment given the available evidence.

Assuming weakly set of weakly annotated set of videos with correct verbs and nouns of a particular domain and we plan to exploit action and object interactions to learning action representations using multiple instance learning, develop novel techniques for action recognition exploiting correlations between the pose and hand gestures and object presence or absence and final temporal annotation of videos using the action/object representations and dynamics learned with the aid of textual descriptions.

In this work we specifically explore how one can leverage external textual descriptions such as transcripts, plot summaries, cooking recipes or craft instructions to automatically annotate arts and crafts videos that have been created for a children's broadcast TV show. Applying natural language processing techniques to the textual descriptions can provide us with both semantic and temporal information about the actions in the video we are processing..

This overall approach is summarized in Fig. 1.1. We assume that we are given a video of an arts and crafts show and also a transcript of the words that are spoken. This could either come from closed captioning information, a transcript created by a fan of the show, or the actual film scripts used to record the episode. We then extract the action verbs and relevant objects from the text, filter them using domain knowledge automatically extracted from external knowledge source and then use a previously learned model of our domain to combine this semantic information to interpret the output of visual action and objects detectors that have been applied to the input video. The ideal output of our system is then a time aligned annotation of the input video (see Fig. 1.2).

Semantic information can be extracted by parsing the phrasal constituents of each sentence to determine the type of action and human interaction via the presence of objects, instruments, and other contextual information. One can

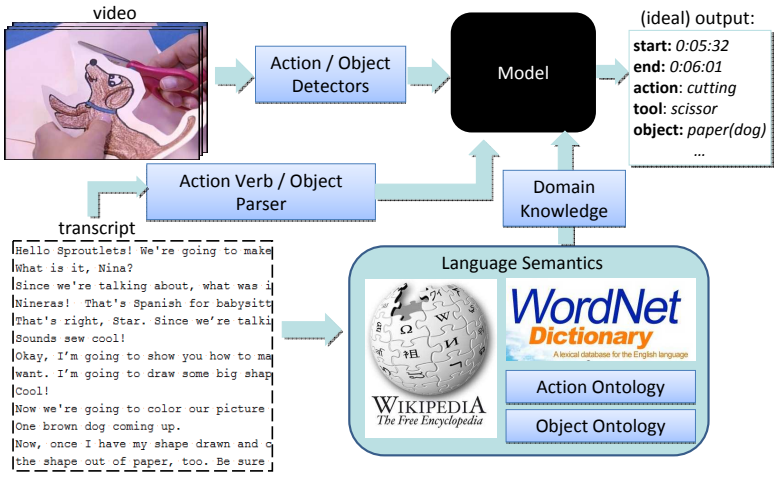



Figure 1.1: General Approach

extract from language the appearance and functional properties of objects and their spatial, temporal, and semantic relationships (e.g. via adjectives, adverbs, prepositions). The language descriptions allow us to identify different entities, varying from named – e.g. persons, characters, etc... to unnamed entities – e.g. locations, instruments, action type, etc.

The textual description can also provide us with temporal information and constraints, such as in what order are the actions happening and if time-aligned text is available (e.g. closed captions or speech recognition) we can actually closely constrain the temporal presence of an action.

## 1.4 Contributions

As part of our efforts we created a new baseline data set (see Figs. 1.3 for some examples) for research into recognition of complex manipulation actions which we hope will be used as a benchmark data set for future research. We were able to license 27 premium broadcast videos from PBS Sprout which were manually annotated. The annotations included the presence of actions, the locations of objects and tools of interest, the time and type of shot transitions, as well as the camera view point.

We also created an end-to-end system that autonomously annotates real-world broadcast videos with the presence of actions and objects. Both the

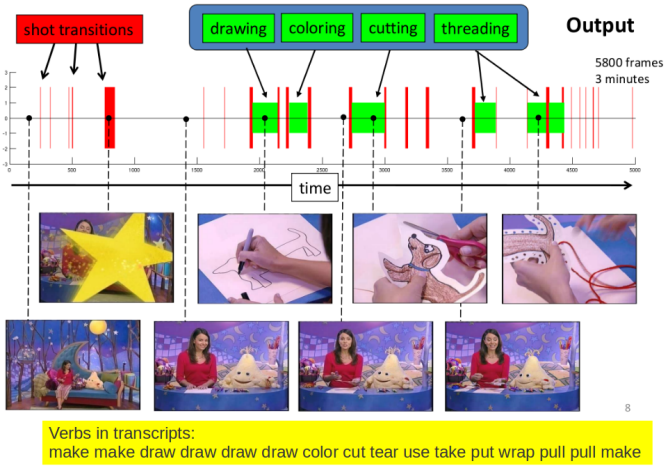



Figure 1.2: Example video "Babysitter's Animal Sewing Cards", PBS Sprout TV with desired annotations (shot boundaries and action labels)

dataset and the code will be publicly available from the workshop website, thereby reducing the barrier of entry for further research.

In the following chapters we will describe the algorithmic approaches we explored and devised, as well as the results we achieved. The overall architecture of the system consisted of the following steps as detailed in the flow chart shown in Fig. 1.4.

First we will describe the natural language processing steps. The descriptive text, we looked at show transcripts as well as associated instructions downloaded from the web, was passed to the information extraction module (Chapter 2) where action verbs and the associated objects and tools are automatically extracted from the text. The extracted (verb,object,tool)-tuples are then pruned and refined using domain knowledge which is automatically learned from external knowledge sources such as Wikipedia, WordNet and search engines such as Google and Yahoo (Chapter 3).

During the preprocessing stage of the video pipeline, the input video is first segmented into semantically meaningful shots and clustered according the presence of human faces and their size using off-the-shelf algorithms for shot boundary detection ([54]), face detection and recognition (Pittsburgh Pattern Recognition), and visual clustering using visual words ([86]). Since these techniques are external software solutions, we will not further describe them in this report. During the next stage the individual video frames are analyzed for the presence of actions, objects, and human hands. We specifically look for human hands because we are specifically interested in manipulation actions, and thus are in-


Figure 1.3: Example videos from PBS Sprout Crafts Data set

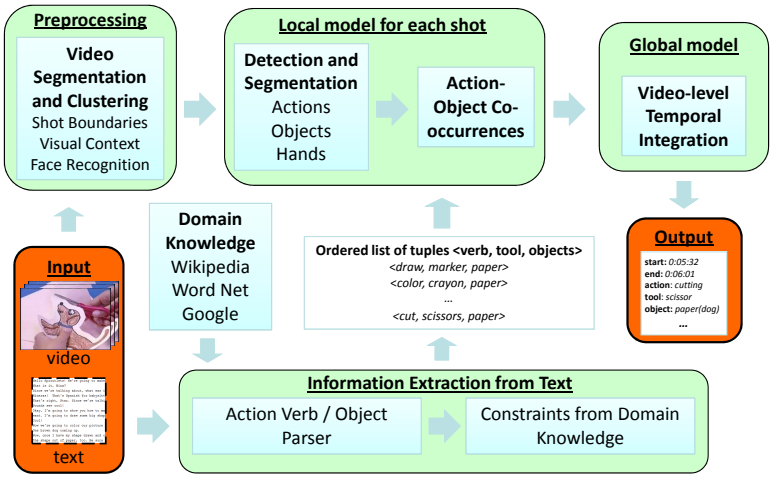



Figure 1.4: Flow chart of algorithmic approach

| | |
|---|---|
| Single Shot Action Recognition using STIP (SVM) | 0.42 |
| previous + Tool + Hand Feature | 0.47 |
| Single Shot Joint CRF Model (STIP+Tool+co-occurrence of verb and tool) | 0.51 |
| Sequence Model CRF with temporal constraints extracted from transcripts (pairwise ordering constraints) | 0.52 |
| Previous model with relaxed pairwise ordering constraints | 0.52 |
| Sequence Model CRF with temporal constraints extracted from online instructions (pairwise ordering constraints) | 0.53 |
| Previous model relaxed pairwise ordering constraints | 0.53 |

Table 1.1: Overall action classification accuracy for the Sprout TV Handcraft Show dataset.

terested in those specific objects that are either moving with or are located in relative proximity to a hand in the videos. The exact algorithms used are described in Chapters 4 (Action Recognition), 5 (Object Detection) and 6 (Hand Segmentation).

At this point now, for each shot a candidate set of actions, tools, and objects has been detected, as well as a list of action verbs and nouns, and their likelihood of their co-occurrences has been computed. We combine all this information into a local model using a CRF as described in detail in Chapter 7. Up to now we have treated each shot as being independent of all the others. During the last step, we take advantage of the fact that we know in which order the verbs and nouns appear in the descriptive text, and that this order should be reflected in the order of the actions that we recognize. Thus, we learn a global CRF model that uses these ordinal relationships as additional features to improve the accuracy of the action annotation (Chapter 8).

In our work we demonstrated how non-visual semantic and temporal information can be integrated with visual information to improve action recognition. In the following Table 1.1 we collected the results we get when we include more and more information. The results show that the information we automatically extracted from text and unstructured domain knowledge is indeed helpful and improves the accuracy of our action annotation system.

Using a local (single shot) model consisting of a support vector machine (SVM) classifier and only spatio-temporal image descriptors as input features, we achieved an overall recognition rate of 42%. We were able to boost the recognition rate by augmenting the SVM feature space using detection likelihoods for different tools and hands to 47%. Integrating the co-occurrence constraints that we automatically learned from text into the local model again increases the recognition rate to 51%. By integrating the global temporal constraints extracted from transcripts and online instructions into a global model we are able to get the best recognition rate ( 52% and 53% respectively).

In summary, during this workshop we successfully demonstrated that the integration of visual and textual information can lead to better action recog-

nition results, and that the relevant domain information necessary to fuse the two information sources can be learned automatically by using freely available data sources on the web. Our hope is that the ideas, code and the dataset that we developed during this workshop will offer a good starting point for the community to continue and build upon this work in a new and exciting research area.

Finally, we want to thank the organizers of the workshop for the great work they did hosting us, and without their tireless work to assemble funds and allowed for this workshop to be so successful. We also want to thank Y. Aloimonos, G. Hager, R. Vidal, and S. Khudanpur for their instrumental help during the planning phase of the workshop.

# Part I

# Natural Language Processing

As said in the introduction, NLP in this project has been used to ground temporal and semantic information in video processing. Among the approaches that are available, we decided to select a simple methodology that could be easily used with any algorithmic framework, that of Information Extraction (IE). In NLP, the role of Information Extraction (IE) is to extract structured information from unstructured documents and this is usually template driven, i.e. find who, what, where, etc... In the following sentence, one can identify the following entities: "Yesterday, New-York [location] based IBM Inc [organization]. announced their acquisition of Bank of America [organization]".

The goal of IE is to allow computation to be performed on unstructured data via the identification and the recognition of entities. We adopted this approach and extended it to an Enriched Information Extraction that is able to capture more complex relationships, such as verbs (to capture action types), objects (to capture what is acted upon), instruments (the tool that is being used), and location (to guide the contextual recognition). We capture also syntactic structures such as verb-object relationships. We allow logical reasoning to draw inferences based on the logical content of the input textual data.

The work that was pursued here extends in two different directions:

1. syntactic parsing to process and understand textual data such as transcripts; we used the Stanford probabilistic parser for dependency relations as well as adapted the Stanford Named Entity Recognizer (CRF) for our task.

2. semantic processing to determine the semantic relatedness between word relations, such as verb-object and object-intrument. We used statistical measures of lexical collocations and lexical similarities and build matrices of co-occurrences to feed action recognition.

The following chapters will develop these 2 aspects.

# Chapter 2

# Information Extraction from Text

## 2.1 Introduction

As has already been discussed, the task of action description is context-dependent: that is, the environment in which an action is done, as well as domain words appropriate to that environment, heavily influence the description of an action. When presented with an instructional video and the corresponding transcript, we need to strategically use the transcript to help the action detection[1].

How exactly can language help? First, the transcript contains a lot of useful information. The first category that comes to mind is that of syntactic information. This syntactic information can tell us the person doing the action, what he or she may be using to complete the action, and what objects the action entails. Note though that to get reliable information regarding the actual action, we need to examine the semantics — and sometimes even the pragmatics — of the transcript. However, if we know the domain/environment of the transcript we may reasonably assume that initial information linking the verb to an action can be derived from syntactic information. For instance, while there may be many different ways to cut something in general (such as with a chef's knife when cooking, or with scissors in a craft show, or with a saw in a home improvement show), in any particular domain it is reasonable to assume that there are a very few, predetermined number of ways to complete the action. Further semantic and pragmatic analysis could be helpful, but the syntactic information regarding verbs can provide a good base.

The second way in which language helps is by providing an ordering of the actions. Ideally, the text would be time-aligned (as in closed captioning) to the video, so that exact start and end points could be reported. However, even without time-aligned text, we may get, at the very least, a partial, relative

[1] Note that while the converse is true (video data can be used to help extract information from text) such a task was beyond the scope of this project.

ordering of the actions. This can be very useful if given to an appropriate global temporal model. Specifically, the language can provide seed information for targeting certain actions, objects and tools all within a relative temporal framework.

Throughout this paper, we will refer to the terms "semantic density" and "semantic relevance" . Semantic relevance is defined at the word-level: we say that a word $w$ is semantically relevant if it is meaningful to and useful for performing the desired task. A sentence or clause is deemed semantically dense if a significant portion of the words in that sentence are semantically relevant; we may define semantic density similarly for a document.

There are two points to make regarding semantic density and relevance: first, both are dependent on the domain and the task at hand. For instance, a document that is semantically dense for one task (e.g. for a general question-answering or paraphrasing system about that document) will not necessarily be semantically dense for another task (e.g. extracting action-related information from that document). Second, deciding semantic relevance and ranking semantic density is inherently subjective. One may try to arrive at metrics to precisely and rigorously quantify the definitions, but any such metric will be heavily influenced and affected by the first issue mentioned. These terms do however provide a way of comparing (more abstractly and generally) two different documents or data sets and so will be used in such a way throughout this chapter.

This chapter will progress as follows: in Section 2.2, we will discuss the gold-standard annotation creation and evaluation metrics for both the main Sprouts data set used in the overall project, and for a newly acquired data set used specifically for the task of extracting action information from text. In Section 2.3 we will discuss the primary linguistic tools we used to complete this extraction, including how we adapt a named-entity recognizer (NER). We will (informally) compare other linguistic tools we could have used and discuss the reasons we used the tools we did. In Section 2.4 we sample some of the past work done that is related to our task. Armed with this history, we then describe the main algorithm we used to extract actions, as well as versions leading up to the final one. Prior to presenting the results for these algorithms, it is most helpful to run some diagnostic tests on the NER to get a sense of its overall performance. In Section 2.6, we discuss the results of the algorithms presented in Section 2.4. As in all projects, there is never enough time to explore all the possibilities, and so in Section 2.7 we discuss some of these possibilities for future work. In Section 2.8 we summarize our work done and presented in this chapter.

## 2.2 Corpora

In this section, we explain the two corpora we used. The PBS Sprouts Crafts corpora (Section 2.2.1) is the primary corpus used in this project and system, and so will be familiar to the reader. However, there are aspects of that corpus

which are unique to this task, that is, unique to extracting action-object-tool triples from the transcripts. In Section 2.2.2, we describe a data set newly created for this project. In both sections, we discuss the gold-standard annotation process along with our methods of evaluation.

### 2.2.1 PBS Sprouts Crafts

As the reader will recall, the primary project data set consists of 27 PBS Sprouts Craft where both the video and the transcript are available. There are generally 40 sentences per show. The transcripts are generally of very high quality and contain few typos. They are also very structured in that each show is a dialog between the same two characters (the host Nina and her co-host Star). As the Sprouts TV shows are made for children, one might think it reasonable to expect a somewhat limited vocabulary along with fairly simple sentence structure. Since these shows are "how-to" by nature, we might also assume that the shows are semantically dense.

Unfortunately, since the show is a dialog by nature, they are semantically sparse (evidence of this may be found in Section 2.5.5). While the two hosts try to verbally engage the viewer, the discussion is generally irrelevant to the task of extracting action-object-tool triples. The shows are much more narrative in nature rather than imperative. This has the consequence of not guaranteeing simple sentence structure: though some instructions are very clearly and simply stated (such as "Cut the paper"), others are hidden within the narrative. Further, since the shows are in the arts-and-crafts domain, determining the richness of the vocabulary is a bit difficult, as nearly anything can be used in a craft: a rock, a coffee filter, an egg shell, a bottle cap, etc. Although the use of synonyms may be limited, the set of possible objects used is extremely large.

Were the transcripts simply instruction sets, we could be assured that every intended action was specified in the text. However, since the transcripts accompany the video, we have no such guarantee. There are times when actions are performed in the video but not mentioned in the text (or only mentioned in the cursory manner of "And now we do this."), and there are times when actions are mentioned in the text but not performed. Therefore, there is no one-to-one correspondence between the video and text.

Below we describe how the gold-standard annotations were constructed and how we will evaluate our program's performance against them. As we use the Sprouts data in diagnostic tests (Section 2.5.5) it is necessary to mention that the methodology described below is not sufficient. To run the diagnostic tests on the Sprouts data we processed the Sprouts data in the same manner as described in Section 2.2.2.

**Gold-Standard Construction** To construct the gold-standard triples for the Sprouts transcripts, a human annotator viewed each episode and recorded when a given action occurred and how long it lasted, along with the objects (and tools) used and how the human interacted with the objects to accomplish the action. The annotator also listed the relative sequential order of the actions.

| Property | Value |
|---|---|
| Number | 3 |
| Verb | Cutting |
| Objects | Paper, Scissors |
| Description | Hands cut out drawing |
| Start Time | 1:32.0 |
| End Time | 1:40.0 |
| Camera Angle | Full, Close Up |

Table 2.1: Sprouts annotations for the sentence "Cut him out with safety scissors."

| Word | Tag |
|---|---|
| Cut | verb |
| him | object |
| out | prep |
| with | prep |
| safety | tool |
| scissors | tool |
| . | O |

Table 2.2: Web annotations

Please see Table 2.1 for an example annotation of the instruction "Cut him out with safety scissors."

Note that the annotations contain information which cannot be derived from text alone, such as describing the camera position and providing the start and end times of the action. Given time aligned text, the times could be provided, but that was outside the scope of this project. Further, providing the "Description" was outside the scope of this project. For the remainder of this chapter, we will only focus on extracting the number field, the verb field and the objects field. Of course, the number field will be relative and dependent on prior detections; therefore, the exact number is not as important as the relative numbering is.

**Evaluation** Given the fact that there is no one-to-one correspondence between the actions mentioned in the text and the actions performed in the video, two evaluation standards must be employed. For each evaluation standard the recall and precision are calculated. Since there is some repetition of actions in the shows (Nina may say "Cut the paper" in one sentence, and then in the next say "Once you are done cutting the paper...") we are not concerned with duplicate verb-object-tool triples as long as they actually correspond to a valid action. We also consider something a false positive detection if and only if it is clearly invalid. An instance of this occurring is if given the sentence "Color the dog

any color you like," the second "color" is labeled as a verb in a triple. Note that given this counting method, the number of false positives is invariant against either of the evaluation standards, as described below.

The primary evaluation standard is against the gold-standard annotations described above. As these annotations were constructed from viewing each show (rather than reading the transcript), this corresponds to judging *against the video*. This means that we judge the program's output triples against what was seen. However, it is also valid to judge *against the text*: that is, to judge the program's triples against what was actually mentioned in the text. In both cases, the evaluations are performed manually. Although it may not seem completely valid to compare the recall and precision on each evaluation standard (the denominators will be different), it can still provide insight into the strength of the algorithms and program.

### 2.2.2 Crafts from the Web

One of biggest limitations of the Sprouts transcripts is that the number of transcripts is just too small. With only 27 shows (and approximately 40 sentences per show), our ability to apply machine learning algorithms or statistical methods to the data is greatly hindered. We therefore decided to mine from four websites[2] 420 crafts. Whereas the Sprouts crafts were primarily for little kids, the crafts we obtained from the web were intended for a much larger range of ages: anywhere between 3 and 13 years of age. Although these instructions must stand alone (there is no accompanying video), the structure is not always strictly imperative. Particularly, as the intended age range increases, we can expect the instructions to become more narrative. As a result, the semantic density will vary as well. Unfortunately, as this data was mined from the web, noise is introduced in the form of possible typos, localized idioms and different format (some are in itemized lists while others are prose-like).

**Gold-Standard Construction** While we would like to be able to determine whether a clause represents an action, we take the more fine-grained approach of determining whether every word is semantically relevant or not[3]. Although many parsers are available (see Section 2.3.1 for more details), they alone are not sufficient for determining semantic relevance: not every linguistic verb is an action verb, just as some actions may be anaphoric references. As we are interested in extracting action-related information, we are primarily interested in whether a given word represents an action *verb*, an action-related *object* or *tool*, some word modifying either an object or tool (*mod*), a preposition *prep* related to an action, an adverb *adv* describing some action, or some other word

[2] http://familyfun.go.com/crafts/, http://parentinghumor.com/, http://crayola.com, and http://www.amazingmoms.com/htm/kidsart.htm

[3] Technically, we determine whether every token is semantically relevant or not. The tokens are defined by the Stanford CRF NER, as described in Section 2.3.2. However, we will continue to refer to the tokens as words, since there is nearly a one-to-one correspondence between the two.

| Have | your | kids | cut | the | shapes | with | scissors | . |
|---|---|---|---|---|---|---|---|---|
| O | O | O | verb | O | object | prep | tool | O |

Figure 2.1: Example gold-standard annotations for the web-extracted sentence "Have your kids cut the shapes with scissors."

$(O)$ which is deemed as irrelevant to action extraction. To that end, we may define the set of appropriate tags as $T = \{\text{O}, \text{adv}, \text{mod}, \text{object}, \text{prep}, \text{tool}, \text{verb}\}$. Given the tag set $T$ and a document $D$, every word (token) $w \in D$ was assigned a tag $t \in T$. Note that these tags are with respect to action descriptions and so do not necessarily correspond to linguistic concepts. As a concrete example of this, please see Figure 2.1. Notice that although there are two verbs in the sentence ("have" and "cut"), we are only interested in "cut"; in this case "have" is rather weak when it comes to describing actions.

Due to time constraints and for consistency, the tags were assigned only by one person. While using multiple annotators is preferable, one major issue is that some words may reasonably have multiple tags. For instance, in the sentence "Using glue, paste the shapes," should "glue" be tagged as an object or as a tool? It is extremely difficult to give a "correct" answer in this case, as glue may be applied by a glue stick or bottle (in which case it would be better to tag it as a tool), but glue may also be applied by using a brush (in which case it would be better to tag glue as an object). For this reason, we allow every word (token) $w$ to map to a non-empty set $T_{\text{gold}} = \{t \mid t \in T\}$. In all 420 crafts, there are 58,399 tokens, but only 208 of them have more than one plausible tag. For the most part then mapping to a set of plausible tags does not pose a problem.

As mentioned in Section 2.2.1, the Sprouts data were also tagged in this manner. An example is given in Table 2.2. To annotate/construct gold-standards for both the web data and the Sprouts data according to this style, a simple boot-strapping procedure was used. Details may be found in the documentation.

**Evaluation** For the web-based data, we are concerned with individual word tags and so we simply have a multi-class classification problem. Therefore unlike the evaluation described in Section 2.2.1, the evaluation for the web-extracted data may be done automatically. For any given binary classification, we may use the following confusion matrix,

| | | Gold Standard | |
|---|---|---|---|
| | | **True** | **False** |
| Prediction | **True** | A | B |
| | **False** | C | D |

and calculate the **accuracy**, defined as $\frac{A+D}{A+B+C+D}$; the **recall**, defined as $\frac{A}{A+C}$; and the **precision**, defined as $\frac{A}{A+B}$. We examine eight binary classification problems: seven are of $t$ vs. all other tags, for every $t \in T$, and the eighth is whether a word was correctly labeled as semantically relevant, even if its

predicted tag did not correspond with the gold standard tag. Since we allowed the mapping between words and tags to be multivalued, we simply check if the predicted tag $t_{\text{predict}} \in T_{\text{gold}}$.

Note, if we made $T_{\text{gold}}$ an ordered set, where the order of the tags in $T_{\text{gold}}$ directly corresponded with how correct a given tag was, then we could apply the (slightly) more strict evaluation of checking whether $t_{\text{predict}} = T_{\text{gold}}\,(1)$. That is, simply check if $t_{\text{predict}}$ is the same as the first entry in $T_{\text{gold}}$. Given that only 3.5% of the tokens in the entire web-based data set have more than one plausible answer, one would not expect the results to change dramatically. This is verified in Section 2.5.

Alternatively, interpreting $T_{\text{gold}}$ as an ordered set, one could attempt to arrive at some weighting scheme where a prediction $t_{\text{predict}}$ was considered correct if $t_{\text{predict}} \in T_{\text{gold}}$, but was given a penalty based on how removed $t_{\text{predict}}$ was from the initial position in $T_{\text{gold}}$. That introduces a whole plethora of issues, namely determining the penalties. Depending on the context, some tags may be more plausible than others and so any weighting scheme should reflect that. Determining these penalties is outside the scope of this project, however, and so this option was not pursued.

One important distinction to note between the evaluation just described here and the evaluation described in Section 2.2.1 is that the evaluation here is word/token-based, whereas the evaluation previously described is action-based.

## 2.3 Primary Linguistic Tools

As described in Section 2.1, natural language text contains a lot of useful information; and as we are exploring in this chapter, the key is how to automatically extract that information. Since we are concerned with verb-object-tool triples, the most direct way is to use a syntactic parser. In Section 2.3.1 we describe the parser we use, and in Section 2.3.2 we discuss the limitations of the parser and how we attempt to overcome them.

### 2.3.1 Probabilistic Parser

The task of syntactically parsing a sentence is one with a very rich history, to which we cannot hope to do justice in this chapter. For an excellent introduction to the tomes written on this topic, please see [4]. Similarly, while there are too many available parsers to provide an exhaustive list, we will briefly highlight a few parsers which have had success. One such parser (MINIPAR, [55]) has been used in other successful systems, such as in [56]. Another such parser is MontyLingua [57] which attempts to use "common sense" in the parsing task. Unfortunately, [57] does not make the dependencies easily available. While [55] does provide the dependencies, we decided to use the Stanford probabilistic parser, as described in [40, 41], since it was extremely easy to set up and it also easily provides the parse tree. Although we did not end up using the parse tree, initially we were unsure of whether we would use it or not and did not want

to restrict ourselves. Needless-to-say, the Stanford parser provides the part-of-speech tags for a given parse of the sentence and the dependency relations. It also makes it very easy to access the $k$ best parses for a sentence. All of these options made the Stanford parser very appealing to use.

### 2.3.2 Adapting a Named Entity Recognizer (NER)

However, every parser has its advantages and disadvantages. We found out that one disadvantage of the Stanford parser was its inability to handle imperative sentences correctly. For instance, given the sentence, "Once that is done, tape or glue the pieces...," neither tape nor glue would be labeled as verbs; rather, they would each be labeled as nouns.

As mentioned above, we did have the option of examining the $k$ best parses. When we did this, the parser generally correctly parsed imperative sentences within $k \leq 3$. However, doing so caused some nouns to be labeled as verbs, hence introducing false positives. Without more knowledge, it could be quite difficult to distinguish between any true positive and false positive verb detections.[4]

We examined other parsers, such as [57, 55] but the results for imperative sentences/clauses were largely the same as with the Stanford parser. It is not entirely surprising that many of the statistically trained parsers cannot handle imperative sentences well since they are primarily trained on professional news data (from sources such as the Wall St. Journal, for example). We would not expect many news stories to contain imperatives and so cannot expect parsers trained on the news data to perform well on all imperative sentences.

If we make the assumption that imperative clauses directly correspond with an action (especially in instructions sets, which both of the corpora are), then it is reasonable to hypothesize that we do not need to do a full syntactic parse[5]; we just need to extract certain pieces of information. It suffices then to create an action/imperative specific part-of-speech tagger. Note how this is very similar in nature to named entity recognition, except instead of extracting or recognizing named entities, we are recognizing action-related events (and associated objects). Given the remarkable ease with which one can train it, we decided to use the named entity recognizer (NER), implemented as a conditional random field, made available by Stanford ([32]). As it has been demonstrated that a CRF can perform very competitively as a POS tagger, when compared to other probabilistic models ([45]), we hypothesized that an NER could help in recognizing patterns and so would be well suited to our task. Empirical evidence supports this hypothesis, as discussed in Section 2.5. Although we are technically not using the Stanford NER ([32]) as a named-entity recognizer, but more

[4] We could have employed some weighting scheme, similar to that described in Section 2.2.2, to try to handle the multiple parses. There was not enough time to accomplish this and so could be pursued as future work.

[5] Not to mention that doing so could be extremely costly. The new data would have to be annotated by multiple linguistic experts to ensure consistency. There was not enough time to attempt this. However, as will be discussed, the point regarding the need for a full syntactic parse for imperatives is moot.

as an action-related part-of-speech tagger, we will still refer to it as an NER. The details regarding the training of the NER can be found in Section 2.5.

## 2.4 Algorithms

In this section we document the primary algorithm used in the project itself. Before that though, in Sections 2.4.2 and 2.4.3, we document and describe the preliminary systems VERSION1 and VERSION2 of the primary algorithm, where the latter uses the Stanford parser (Section 2.3.1) to build upon the former. Then in Section 2.4.4 we describe the main (primary) algorithm, VERSION3, which is built upon VERSION2 but incorporates the NER as described in Section 2.3.2. First though, we discuss previous and related work in Section 2.4.1, and discuss how our task and data set are different from what others have done.

### 2.4.1 Related and Previous Work

The key challenge of this portion of the project is to be able to automatically generate the verb-object-tool tuples. As has been alluded to previously, this is very much akin to information extraction, although it would be better described as enhanced information extraction. While the general idea is the same (extract the who, what, when, and where) for actions, we wish to incorporate additional linguistic information. Further, to make the process truly scalable, we wish to extract more than just standard named entities, locations, etc., as evidenced in Sections 2.2 and 2.3. As a result, we must look beyond the standard approaches of information extraction.

To perform this enhanced information extraction we would ideally like to extract a deep semantic understanding, along with the syntactic information. The canonical approach has been to augment the grammar with $\lambda$-reduction rules to achieve a first-order logic representation [4]. Unfortunately, not only is this approach expensive, it requires experts to successfully augment the grammar. One may try to learn dependencies relations for creating inference rules [56], although the results can vary quite dramatically even with 1 GB of data. Especially recently, there has been a lot of work to try to automatically learn semantic augmentations: for instance, [35] provides a framework for an augmented CFG which on certain domains (such as parsing commands given in a chess game) works well. There has been a successful attempt to perform unsupervised semantic parsing on biomedical abstracts [72]. [53] presented a generative model that has been shown to yield increased performance on three different data sets [53].

However, our task, with respect to our domain, is significantly different than the above mentioned previous work. With only 27 three-minute transcripts, we do not have nearly enough data to apply a DIRT-algorithm to our task. There is also the underlying theme among the above work of relatively high semantic density: by examining commands given in a chess game (such as "Move the knight to B8"), [35] can be relatively assured of high semantic density in

their data (given the concentration involved in chess, one would not expect a lot of semantically-irrelevant data). Similarly, by using biomedical abstracts, [72] have made the decision to restrict themselves to a set with high semantic density. [53] test on three different data sets, but all of those have relatively high semantic density, especially compared to the Sprouts data. Although the semantic density issue may not seem to be an incredibly serious issue, we briefly examine and analyze it throughout Section 2.5. While working toward a full semantic analysis would be more appealing, the low semantic density could result in the costs involved in creating semantic rules (such as augmenting grammars) far outweighing the benefits. The size of the data set also presents challenges to using established and successful techniques. Given this, we decided to pursue and provide our own algorithm, demonstrating what is needed to successfully extract action-related information from our text.

### 2.4.2 Phrase-Based Extraction

Given the challenges facing us, such as very low semantic density (page 13), a new data set and a fairly untackled task, we decided to start with the most basic, most naive algorithm we could think of: simple pattern matching. For instance, pattern matching and variants thereof have been used to some success [39], though we had no intention of doing anything as complicated as [39]. Rather, we simply wanted a rather basic baseline, which we will call VERSION1.

To that end, we hypothesized that given we were working with children's shows, the host would want to be very engaging and so might be more likely to lead the viewers into the next instruction such as in "And now we're going to ⟨*do-some-action*⟩,"); or prompt instructions via modals, such as "After this is done, you can ⟨*do-some-action*⟩." From the very beginning of planning this algorithm, there was no intention of using it in the final project so we tried to match four different phrases, and hence did not try to make this option very competitive. Though it may have been possible to finely tune VERSION1 so as to get both high recall and precision, intuitively it would not scale well to other domains (such as cooking shows, home improvement, or even other craft shows). One of our objectives was to design a scalable algorithm and spending time to refine VERSION1 would have been counterproductive to that.

### 2.4.3 Seed Words + Phrase-Based Extraction

One of the main limitations of VERSION1 is that it uses no linguistic knowledge — despite us having access to it (Section 2.3), and only minimal domain knowledge — if one wishes to count matching four generic phrases as "domain knowledge." To proceed in a scalable manner, we wanted to see what we could do with varying levels of both linguistic and domain knowledge.

There are many ways one can have domain knowledge; one of the first that comes to mind though is if one had access to a list of accepted verbs used in the domain. Therefore for our next implementation, VERSION2, we first assumed that we had access to a list of seed craft action verbs. To generate the list for

this bag-of-words approach, we simply thought of acceptable craft verbs (such as "{cut}, {glue}, {color}"), and then, as a slight simplification, age-appropriate synonyms (such as "{snip, tear}, {paste, attach}, {shade, brush}")[6]. While this may initially resemble a standard bag-of-words approach, if we used no other knowledge, the system would predict that some nouns would be actions (for instance, both cut and color can be verbs and nouns). To incorporate linguistic knowledge, we obtained the POS tags and dependencies from the best parse of each show provided by the Stanford parser (see Section 2.3.2 for a discussion of the best parse returned versus the $k$-best parses returned, and why we chose only to use the parse with the highest probability).

The algorithm for VERSION2 proceeded as such: for every sentence in the transcript, we would try to match the same four action-introducing phrases as described in Section 2.4.2 for VERSION1. However, it also extracted those lines and clauses that contained one of the predefined action seed words. It then used the POS tags and parse dependencies provided by the Stanford parser to verify that the found action word was actually a verb and to extract the direct objects and (sometimes) the implements.

### 2.4.4 CRF + Seed Words + Phrase-Based Extraction

The success (or failure) of VERSION2 is almost entirely dependent on [(1)]

1. the list of seed words provided, and

2. the output of the parser . However, as noted in Section 2.3.2, the parser has trouble when it comes to imperatives. As the Sprouts data are, first-and-foremost, instructional videos, the inability to handle imperatives could certainly pose a problem. As described in Section 2.3.2, some of the most successful and current parsers lack the capacity to handle imperatives. This need to compensate for the parser lead us to the idea of using an NER as an action-related POS tagger, as described in 2.3.2.

We must allow the possibility for the parser and the NER to tag some words incorrectly. However, having hypothesized that adapting the NER, and training it specifically to look for action-related information in text, would act as a secondary part-of-speech tagger, we would hope that the parser and the NER could work together and complement each other. Therefore we simply need to find strategic ways to use the NER (or rather, its output on a given show) in the algorithm for VERSION2. The algorithm is given on page 24 in Algorithm 2.1.

First and foremost, we want to use the NER output to compensate for potential parser errors . We may also use it to verify verbs (increase or decrease the system's certainty/confidence that a detected word is actually an action verb), and also add seed words to the bag. The issue of verifying verbs and the certainty measure is discussed below. We did not incorporate using the NER to

[6]There are obviously synonyms that we did not use. That was because we deemed those verbs too complex/sophisticated for a children's show. This simplification did not seem to harm the performance; indeed, including them would have resulted in many useless computations.

add verbs to the bag, though it would be very simple to implement and feasible, as will be discussed in Section 2.5.6.

We also use the CRF output to help develop heuristics or certainty measures to say how certain it is that a returned verb-object-tool tuple is actually an action. Currently everything is based on the verb $v$ in the verb-object-tool triple (and the stemmed version of $v$, $v_{\text{stemmed}}$), and the certainty measure is just a linear combination of four features $\mathbf{f}$:

[(1)]

1. whether the Stanford parser says the verb $v$ is a verb ($f_1 \in \{0, 1\}$),
2. whether the CRF says the word is a verb ($f_2 \in \{0, 1\}$),
3. whether the CRF says the word is semantically relevant to action detection ($f_3 \in \{0, 1\}$), and
4. the frequency count (normalized to 1) of $v_{\text{stemmed}}$ in the CRF training data ($f_4 \in [0, 1]$).

Features 1, 2 and 3 are binary, and feature 4 is a real number between 0 and 1, inclusive. To calculate $f_4$, if $v_{\text{stemmed}}$ has been seen by the CRF before, then simply divide the number of times the CRF saw $v_{\text{stemmed}}$ by the maximum count of all stemmed verbs seen by the CRF; otherwise if $v_{\text{stemmed}}$ has not been seen by the CRF before, just set $f_4 = 0$. Using four weights $\mathbf{w}$, the certainty is just $c = \mathbf{w} \cdot \mathbf{f}$. Although ideally the weights would be parameters to be learned, for this project we have just manually set the weights as 4/13, 4/13, 2/13 and 3/13, respectively. Note that this way $c$ is a real number between 0 and 1; it can therefore be interpreted as a very rough approximation of the probability of $v$ representing a legitimate action. It would also be good if more features were added to it as well, though we leave that to future work.

There is at least one additional possible use for the NER, which we unfortunately did not have time to fully explore. We would have liked to use the NER in a web-crawling/mining fashion: that is, given a list of websites, have the some seeded/minimally-trained NER mine each site for craft instructions. When it found instructions, it would predict on the instructions and add them to the overall training data. One potential problem with this is that the NER would be advising itself in the learning so for the tags and patterns it got correct, it would be likely to get future ones correct as well; but for the tags and patterns it got incorrect initially, it would be likely to get similar future ones incorrect as well. However, if minimal training could seed the NER to do very well, this potential issue could either be acceptable, or require sampled correction of future mined data. While a full implementation of this idea is left to future work, we were able to simulate this web-crawling in Section 2.5.4 with a fair bit of success.

**Algorithm 2.1** Algorithm for VERSION3 of ActionExtractor.

$T$ =transcript of a Sprouts show
$P$ =parse of $T$
$L$ =set of labels for NER as in Sec. 2.3.2
$S$ =break $T$ into sentences, corresponding to those in $P$
$W = \bigcup_{s \in S; w \in s} \{w\}$ //all words in $T$
$C_s : W \to L$ =output of NER on $T$, indexed by $s \in S$
$A$ =seed action-verbs (bag-of-words)
$D$ =predefined dependency predicate names
$P_t : S \times A \to \{0, 1\}$//POS tags from parse of $T$
$P_d : S \to D \times W \times W$ //dependencies from parse of $T$ $\mathbf{w} = (4/13, 4/13, 2/13, 3/13)$ //weights for certainty measure
$\mathbf{f} = \mathbf{0} \in \mathbb{R}^4$ features for certainty measure
foreach $s \in S$
if $s$ matches a phrase, as described in Sec. 2.4.2
Process it as in Sec. 2.4.2
foreach $a \in A$
if $a \in s$ AND $P_t(s, a) == 1$

    $f_1 = 1$
    DirObj=()
    Tools=()
    foreach $(d, a, o) \in P_d(s)$
        if $d \in D$ is direct object dependency
            Append $o$ to DirObj
    foreach $\{(d, a, t), (d, o, t)\} \in P_d(s)$
        if $d \in D$ is tool dependency
            Append $t$ to Tools

if $a \in s$ AND $C_s(a) ==$'verb'

    $f_2 = 1$

if Finding d.o. and tools in above manner fails

    DirObj=$\{o \mid C_s(o) ==$ 'object'$\}$
    Tools=$\{t \mid C_s(t) ==$ 'tool'$\}$

if $a \in s$ AND $C_s(a) \neq$'O'

    $f_3 = 1$

Calculate $f_4$ from $\bigcup_{s \in S} C_s(\cdot)$
Calculate certainty $cert = \mathbf{w} \cdot \mathbf{f}$
print $(a, DirObj, Tools, cert)$

## 2.5 Adapting a CRF NER

As mentioned in Section 2.2, the PBS Sprouts data form too small of a data set to be adequately used for training; further, using the Sprouts data directly for any statistical purposes or machine learning algorithms would create possible train/test conflicts later on in the project. However, we may without fear of conflict use the data mined from the web (Section 2.2.2) for machine learning/statistical techniques. Below in 2.5.1, we describe the various ways in which we partitioned the 420 web-mined crafts. Then in Sections 2.5.2-2.5.4 we experiment with various partitions of the data and report the results. Section 2.5.5 discusses diagnostic tests of the NER on the Sprouts data; that is, we compare the differences between the Sprouts data and the web data.

### 2.5.1 Partitioning of Data

Recall that there are 420 crafts mined from the web. As the creation of a new data set contains a lot of overhead (first in obtaining it and then in constructing the gold standard), we decided to first obtain a fraction of the crafts we could possibly get and see how much the CRF and new data actually affected our system (that is, analyze the difference between VERSION2 and VERSION3). As a result the data arrived in two stages and initially there were only 121 crafts. We therefore set about obtaining an additional 299 crafts. By the time we realized just how beneficial this new data were, there was not enough time to consistently and faithfully annotate the additional crafts. We could not reliably use these new annotated data for additional training, but we still were able to use the plain text of the additional crafts.

As a notational convenience, we will refer to the various classifiers that we trained as **Training-Testing:NumberCrafts**; that is, using **NumberCrafts**, we used (roughly) **Training**% of them for training and **Testing**% for testing. Since we acquired the crafts in two discrete stages, if **NumberCrafts** $\leq 121$, then we used a subset of the initial 121 crafts of size **NumberCrafts**.

Using this notation then, we primarily trained two classifiers: **70-30:121** and **100-0:121**. Specifically, we trained **70-30:121** on 87 crafts and tested on the remaining 34; whereas for **100-0:121** we trained on all 121 crafts. Training this classifier would result in no crafts remaining from the initial 121, so the only experiments for which we used **100-0:121** are described in Section 2.5.5. In all other cases, unless explicitly specified otherwise, all tests were done on the original 34 web-mined crafts.

One of the reasons for using [32] is the ease with which one can train it. Therefore the exact training instructions and procedure are left to the instructions/FAQ found online[7] and the documentation for this project. Also, one may recall the boot-strapping procedure described in Section 2.2.2 to annotate the data.

[7] http://nlp.stanford.edu/software/crf-faq.shtml

| Semantic Relevance for **70-30:121** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc. One (%) | Acc. Mult. (%) | Rec. One (%) | Rec. Mult. (%) | Prec. One (%) | Prec. Mult. (%) | F1 One | F1 Mult. |
| **7 Classes** | 89.10 | 89.88 | 82.15 | 82.96 | 90.33 | 91.59 | 86.04 | 87.06 |
| **3 Classes** | 91.23 | 91.87 | 85.21 | 86.03 | 87.39 | 88.60 | 86.28 | 87.30 |

Table 2.3: Results for **70-30:121** with 7 classes and **70-30:121** with 3 classes, tested on the 34 web-based craft files from the original 121 crafts.

### 2.5.2 Fully Supervised Training

Initially we wanted to get an upper-bound on the performance we could expect from the NER. To do this we trained the **70-30:121** classifier on all 87 training files and tested on the 34 test files. Recall from Section 2.2.2 (page 17) that we could evaluate the results using only one possible answer (so $t_{\text{predict}}$ was equal to the first element of $T_{\text{gold}}$, interpreted as an ordered set) or using multiple answers ($t_{\text{predict}} \in T_{\text{gold}}$). For both of those, we have eight ways of viewing the problem as a binary classification: examining each of the tags individually, and also checking whether the classifier was correct in determining semantic relevance. The results for semantic relevance (was a token correctly tagged as semantically relevant or not, even if the predicted tag was not correct) are given in Table 2.3, in the row "7 Classes" (this is the default).

The first thing one should notice is the difference between evaluating against only one answer versus a whole set of answers (the One vs Mult. columns in Table 2.3); while the multiple answers scheme does perform better than the singular answer scheme, the difference between the two is minimal. The conclusion that using either scheme does not make a very significant difference is mostly verified in all other experiments run on the NER. When it is not, it is typically due to a tag which accounts for a very small percentage of all the tokens. Therefore, after concluding the primary analysis of Table 2.3, we will only focus on evaluating with multiple plausible answers, especially since by evaluating with multiple plausible answers, we do not run into the troubles described in Section 2.2.2.

As mentioned previously, the recall is what is most important to the overall project. It may seem a bit disappointing then that while the precision is very high, the recall is low when compared to the precision. With only 87 training files there may not be enough data to sufficiently learn all the necessary parameters, so we decided to train a new classifier on those same 87 files, but use three possible tags (*O*, *object* and *verb*) rather than the seven used in **70-30:121**. These results are in Table 2.3, in the row labeled "3 Classes." To accomplish this, we used the mapping $\{O, adv, prep\} \mapsto O$; $\{mod, object, tool\} \mapsto object$; and $\{verb\} \mapsto verb$. As expected, this simplification bettered the accuracy; it also bettered the recall. However, it harmed the precision. However, Table 2.3 is only examining one of the eight possible evaluations; Table 2.4 compares the 7-class

| Per-Tag Results for **70-30:121** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Measure: # Tags: | Acc. 7 (%) | Acc. 3 (%) | Rec. 7 (%) | Rec. 3 (%) | Prec. 7 (%) | Prec. 3 (%) | F1 7 | F1 3 |
| *O* | 89.80 | 91.82 | 94.69 | 94.67 | 88.86 | 93.37 | 91.06 | 94.06 |
| *object* | 93.85 | 92.65 | 81.02 | 83.15 | 90.87 | 86.67 | 85.66 | 84.87 |
| *verb* | 97.84 | 97.96 | 87.84 | 88.51 | 87.64 | 87.92 | 87.74 | 88.22 |
| *adv* | 99.66 | — | 55.17 | — | 80.00 | — | 65.31 | — |
| *mod* | 98.98 | — | 60.78 | — | 84.93 | — | 70.86 | — |
| *prep* | 97.72 | — | 81.98 | — | 91.23 | — | 86.36 | — |
| *tool* | 99.38 | — | 72.13 | — | 75.86 | — | 73.95 | — |

Table 2.4: Results for **70-30:121** with 7 classes and **70-30:121** with 3 classes, tested on the 34 web-based craft files from the original 121 crafts.

NER and the 3-class NER under the seven remaining (per-tag) evaluations.

Although recall is generally more important than precision, as will be explained in Section 2.6.3, in dealing with the Sprouts shows we actually want to use the default 7-class **70-30:121** instead of the 3-class (put briefly, this is because our recall is more-or-less optimized and we do not want to hurt the precision). From here-on-out, unless explicitly stated otherwise, any **Train-Test:Number** classifier will be trained and evaluated using seven tags.

### 2.5.3 Semi-Supervised Training

While the results of **70-30:121** from the above section are strong and promising, one potential downside is that it is fully supervised. This could seem to be conflicting with the idea of using the CRF in a web-crawler type of fashion, to become better and gather more domain-related terms (as described in Section 2.4.4. We then decided to explore just how many crafts one needs to train the CRF on before it approaches (or reaches) similar performance to that of the fully supervised CRF, described above.

The underlying algorithm is quite simple: using only the training portion $\mathtt{C}_{121}$ of the 121 crafts (87 crafts in total), for every integer $n \in \{n \mid n_1 \leq n \leq n2, n = k \cdot \Delta n + n_1, k \in \mathbb{Z}_{\geq 0}\}$, for $\mathtt{I}$ iterations we chose a random subset $\mathtt{T}$ of size $n$. We then trained a classifier $\mathbb{C}$ on $\mathtt{T}$. We used $\mathbb{C}$ to predict tags ($\mathtt{H}$) for every craft in $\mathtt{C}_{121} \backslash \mathtt{T}$. We assumed the output was correct, retrained $\mathbb{C}$ on $\mathtt{C}_{121}$ and $\mathtt{H}$ and then tested on the test portion of the 121 crafts. To get statistics, we averaged over all $\mathtt{I}$ iterations. For a more formal algorithm, please see Algorithm 2.2. In our experiments, $n_1 = 5, n_2 = 85, \Delta n = 5$ and $\mathtt{I} = 10$.

The left-hand side of Figure 2.2 compares the recall and precision in the binary semantic relevance tagging measure. And since with the Sprouts data we are most interested in finding action verbs, the right-hand side of Figure 2.2 shows the recall and precision scores for the semisupervised learners, with respect to the verb tag. The straight line in each graph represents the perfor-

**Algorithm 2.2** Algorithm for semi-supervised training of crafts.

```
C_121 = set of 87 training crafts of 70-30:121
G_121 = set of 34 test crafts of 70-30:121
for n = n_1 to n_2 by Δn
    for i = 1 to I
        Get random subset T ⊂ C_121, |T| = n
        Train NER ℂ_1 on T
        H = predictions of ℂ_1 on C_121\T
        Train NER ℂ_2 on T∪H
        Test ℂ_2 on G_121 and record results
```

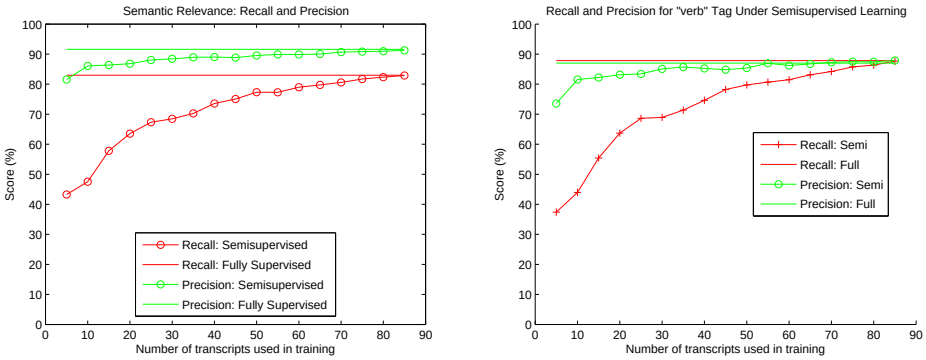



Figure 2.2: Recall and precision comparison for semantic relevance metric using semi-supervised learning. The corresponding fully supervised classifier is 70:30-121.

mance of **70-30:121**, the fully supervised classifier. As expected, all the results converge to the fully supervised learner. Interestingly though, the recall takes a much longer time to converge than does the precision, in both the semantic relevance and the verb classification problems.

Notice how we can get comparable accuracy and precision to the fully supervised learner by using about one-third of the data, but we need nearly all of it (about 70-75 well-annotated transcripts) to get comparable recall. These data indicate that given more training data, the verb recall would continue to increase.

### 2.5.4 Simulated Web-Crawling

The above semi-supervised learning experiments suggest that we can get fairly comparable performance overall even if we do not fully annotate all of the data. We then extended the experiments of semi-supervised learning to include all the training data. We used Algorithm 2.2, with parameters $n_1 = 50, n_2 = 110, \Delta n =$

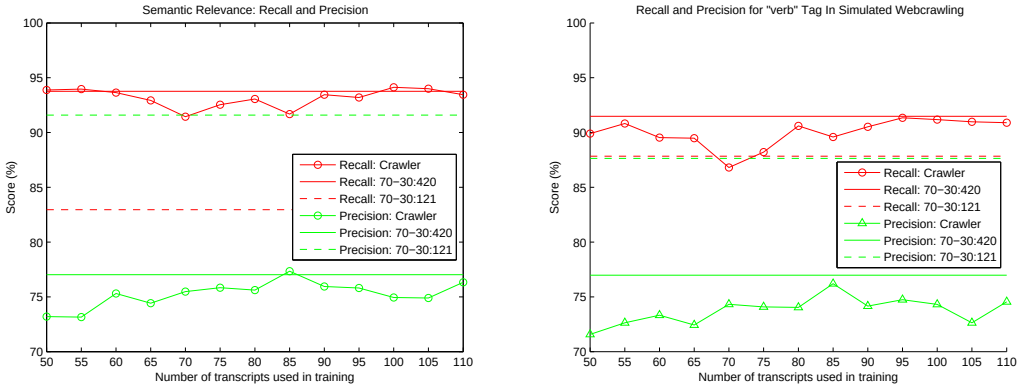



Figure 2.3: Select graphical results for the simulated web-crawling experiments, compared to **70-30:121** and **70-30:420**.

| Classifier Used | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **70-30:121** no extra training | 89.88 | 82.96 | 91.59 | 87.06 |
| **70-30:420** no extra training | 85.83 | 93.76 | 77.02 | 84.57 |
| **70-30:121** predict on all training | 84.27 | 94.18 | 74.59 | 83.25 |

Table 2.5: Results for simulated web crawling, tested on the 34 web-based craft files from the original 121 crafts.

5 and $\mathtt{I} = 3$. We tested on the 34 test files from the initial 121. Interestingly, while our accuracy and precision decrease, our recall increases when we simulate web-crawling, with respect to **70-30:121**. These data indicate that training on a very few number of really well annotated files and then crawling and mining the web for more data (semi-)automatically can result in performance quite comparable to that of training on many more, human-annotated files! That makes the web-crawling idea/program seem quite promising. Please see Figure 2.3.

### 2.5.5 Diagnostic Testing on Sprouts Shows

None of the NERs were trained on the Sprouts data, so it is quite reasonable to suspect that none of the NERs would do as well on the Sprouts as on the web-obtained crafts. The question though is just how big is that difference? Recall that the output from the NER is evaluated by token/word while the output from any of the action extractors described in Section 2.4 is evaluated by action phrase; while we can expect a large performance increase or decrease from the NER on the Sprouts data to a corresponding performance increase

| Classifier Used | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **70-30:121** | 94.83 | 78.39 | 72.97 | 75.58 |
| **100-0:121** | **95.13** | 80.06 | **74.34** | **77.09** |
| **70-30:121** all extra training | 84.39 | **91.34** | 39.53 | 55.18 |
| **70-30:420** | 88.06 | 91.13 | 46.53 | 61.61 |

Table 2.6: Results for semantic relevance diagnostic tests on Sprouts data.

| Classifier Used | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **70-30:121** | 98.63 | 75.93 | 70.49 | 73.11 |
| **100-0:121** | **98.72** | 73.98 | **73.52** | **73.75** |
| **70-30:121** all extra training | 94.34 | 85.92 | 32.13 | 46.77 |
| **70-30:420** | 96.63 | **91.42** | 44.21 | 59.59 |

Table 2.7: Results for semantic relevance of only the 'verb' tag on Sprouts data.

or decrease from the action extractor, the same cannot necessarily be said for small performance changes. This is due to the differences in evaluation, but also because we use additional information in the extractor (at least in VERSION2 and VERSION3).

As compared to the web corpus where there are nearly 59000 tokens, there are only 13727 tokens/words in all 27 Sprouts Craft transcripts. However, the biggest potential issue seems to be the semantic sparsity of the Sprouts data. This is evidenced by the "O" tag comprising nearly 90% of the entire corpus; compare this to the "O" tag comprising roughly 60% of the test web data. The first metric we will examine is how various NERs do with regard to semantic relevance. The results are given in Table 2.6. In addition to testing the standard **70-30:121**, we also trained and tested a **100-0:121** classifier (Section 2.5.1). These are arguably the most stable and reliable classifiers to use. However, to see just how well the web-crawling idea could actually be adopted, we decided to test **70-30:121** augmented with all the extra training data (as described in Section 2.5.4). Given the similarity between this "web-crawling" classifier and the less reliable **70-30:420**, we also tested the Sprouts data on the latter.

As evidence in Table 2.6, the above claim that **70-30:121** and **100-0:121** are the most stable of the four tested warrants merit. Between those two classifiers, as we provided more consistent training data to the classifier, all of the measures increased. With more reliable training data, we would hope that **70-30:420** would have followed that pattern, but unfortunately it decreased the F1 score (though as above, significantly increased the recall), just as it did in previous experiments.

Similarly, we may examine any of the other seven binary classification problems for these classifiers and the above comments will generally still be valid. For sake of space and relevance to the overall task though, we will show only

the results for classifying the 'verb' tag in Table 2.7. Notice how **70-30:121** yields strong performance, though **100-0:121** is the strongest overall.

### 2.5.6 Additional Benefits of the NER

Obtaining the new craft data set from the web enables us to use the CRF NER as an action-related POS tagger, but it also provides a whole bunch of data which can be analyzed statistically. For instance, one of the primary downsides of VERSION2 is that it relies on a bag-of-words approach. However, examining the verbs in all the data used for the CRF provides additional supporting evidence. For instance, the bag-of-words used in VERSION2 contains 49 stemmed words, whereas we get 205 stemmed words just from the training portion of the 121 files. Of those 49 in the bag, there are three words that **70-30:121** legitimately fails to have seen as verbs in its training set. Using all of the training data though results in all of the verbs in the bag as having been seen. Therefore, the problem of generating the seed words (the bag) has been solved.

We can also extract frequency counts for the verbs seen in the training of the CRF (or if we extend it, to the auto-extracted data from websites). For instance, the top 26 most commonly seen verbs in the 70-30:121 training data were (from most common to least): use, cut, attach, place, tie, make, glue, add, paint, fold, press, draw, apply, decorate, cover, bend, turn, secure, roll, remove, fill, dip, create, tape, hang, form, wrap. How frequently a verb had been seen in training/extraction has already been used as a very simple heuristic for extracting actions from the text, as described in Section 2.4.4; the results are discussed in Section 2.6.3.

The NER indicates that it is scalable to other domains as well. First, we have already seen initial evidence in Section 2.5.4 implying that the idea of using the NER to crawl the web for more data has a lot of potential, especially as it increases the recall (and is on-par with potentially inconsistent human-annotated data!). In annotating the web data (Section 2.2.2), there was nothing specific to the craft domain, but rather only to *actions*. There is not strong evidence, then, to suggest that the same process cannot be applied to other data involving instructions — such as cooking shows, home improvement, etc. Although there was not enough time to fully and accurately test these hypotheses, they remain promising and are definitely viable options for future work.

## 2.6 Results on Sprouts Shows

The results for each of the three versions of the action extractor can be summarized very concisely, and is done in Table 2.8. As expected, as the version number increases, the action recall increases, under both evaluation metrics (by video and by text). However, as discussed in each of the subsequent section these statistics must be interpreted knowing that there were 127 actions that could have been detected in the against-the-video metric, while there were 137 actions that could have been detected using the against-the-text metric. In the

| | Recall (%) | | | Precision (%) | | |
|---|---|---|---|---|---|---|
| **Evaluation ↓ VERSION→** | **1** | **2** | **3** | **1** | **2** | **3** |
| To Video | 37 | 85 | 92 | 88 | 88 | 65 |
| To Text | 46 | 85 | 99 | 90 | 89 | 69 |

Table 2.8: Results of VERSION1, VERSION2 and VERSION3.

rest of this section, we analyze the successes and failures of each version, and look at how each could potentially be improved. In Section 2.6.1 we look at VERSION1; in Section 2.6.2 we look at VERSION2; and in Section 2.6.3 we look at VERSION3.

### 2.6.1 Phrase-Based Extraction

Unsurprisingly, VERSION1 did not yield extremely strong results as evidenced by 35% recall and XX% precision, evaluated against the video. Recall that this was simply matching four action-introducing phrases, and so would be useless for a sentence such as "A grown up sprout can help you with cutting out your construction paper, or you can tear the paper carefully!" While it is arguable that such a phrase could have been included in the matching process, doing so would have greatly diminished both the robustness of the system (we can easily come up with action introducing phrases which would require either a lot of creativity to handle in a general case or a lot of hand-tailoring) and how well it generalizes to other domains and tasks.

There are technically three main areas in which VERSION1 is weak: the inability to handle general instructions/phrases, the inability to handle imperatives and the inability to handle anaphoric description. As just discussed, though, the first issue could theoretically be handled if enough time were spent to hand-tailor cases. The latter two weaknesses are ones which we consider too difficult to handle without additional knowledge or information. For instance, how can imperatives be handled with neither linguistic information nor domain information? The third disability mentioned is anaphoric action description. As an example of that we mean that a cutting action may be described in the text solely by the sentence "We're now going to do this." Except for detecting that some action is taking place, based solely on the text (i.e. with no visual input) there is no easy way to figure out what might be occurring. Doing so would most likely require semantic, and hence linguistic, analysis. Interestingly though, as anaphoric description tended to be included in our action-introducing phrases (the ones used for VERSION1) we were able to detect the presence of these descriptions: they were not just entirely helpful if one wants a specific action named in a shot, rather than just the fact that some action may be happening. However, anaphoric descriptions affect all three versions and so the issue will be discussed in more detail in the next section.

### 2.6.2 Seed Words + Phrase-Base Extraction

Using both lexical domain information and linguistic information, VERSION2 was able to get much better results than VERSION1. Specifically, against the video VERSION2 yielded 85% recall and 88% precision, while against the text it yielded 85% recall and 89% precision. While we have to keep in mind that these numbers are not entirely comparable, they do indicate that the transcripts present to VERSION2 challenges proportional to the metric used. From the discussions in previous sections (2.6.1, 2.3.2), one can guess that these challenges are primarily handling imperatives and handling anaphoric description. As anaphoric description is not included in the against-the-text evaluation, that would seem to indicate that there are a significant number of imperative sentences that are incorrectly parsed (and hence the incorrect POS tags are returned). Further, since the evaluations were done manually, all of the data were manually inspected, and we were able to see that, save one or two sentences, nearly all of the actions that were not detected in the text were due to imperatives. Here we have empirical evidence indicating that some other information source is needed. The precision will be discussed shortly.

Unfortunately, the against-the-video metric is a theoretically slightly more difficult task for VERSION2. This is due not only to the imperatives-issue, but also because of anaphoric description. When anaphoric description occurs, the only source of information is visual, so the anaphoric description will be included in against-the-video count. Since VERSION1 was able to detect most anaphoric descriptions, VERSION2 could as well. Such detections were counted as valid only if the action that was seen in the video could be obtained from the action-object-tool tuple returned. As an example, if we had the (slightly easier) anaphoric description, "Now we're going to do this with the tape," VERSION2 may detect ⟨`verb:`do; `object:`; `tool:`tape⟩. We would then be able to detect that some action with tape is occurring, which could then in theory be passed along as information to the visual detection portion of the system.

Despite this potential increase in difficulty, the results for VERSION2 against-the-video are nearly identical to those against-the-text: there was 85% recall and 89% precision. Remember though that there were 137 possible actions in the text, versus only 127 depicted in the video. In analyzing this metric, the manual evaluation and inspection of the results are more helpful than simply using the recall and precision numbers to try to reason out strengths and weaknesses. Specifically, the manual inspection indicates two things: although imperatives are still an issue when measuring against the video, anaphoric description is just as much an issue. Since the results for both metrics of VERSION2 are nearly identical, it makes gleaning support for these claims slightly more difficult; it also enables one to become trapped in comparing the two metrics. The results for VERSION3 however provide strong evidence to greatly support this claim.

The precision for both metrics is something that needs to be addressed. As mentioned in Section 2.2.1, a predicted action was considered a false positive detection when it clearly had nothing to do with introducing an action related to that craft. As a result of this definition, the number of false positives was

invariant to the metric used. It is worth noting, then, that although the precision was 88% against the video and 89% against the text, there were only 15 false positive detections across all 27 shows. On average then there were 0.55 false positive detections per show.

### 2.6.3 CRF + Seed Words + Phrase-Based Extraction

We have verified our hypothesis that using linguistic and domain information helps to extract action related information from text. But we have also seen that there are some issues that have not been resolved, such as the inability to reliably handle imperatives and anaphoric descriptions. VERSION3, built upon VERSION2 but with additional information provided by the NER, was designed to fix and better significant parts of VERSION2. The classifier used to generate these results was **70-30:121**. When measured against the text, VERSION3 achieved a 99% recall but 69% precision; when measured against the video, it achieved a 92% recall and 65% precision. In total, there were 62 false positives for a total of 2.30 false positives per show.

Clearly the recall is an improvement from VERSION2 under both metrics. For instance, against the text, VERSION3 missed only one action in all shows. This one action was a 'rare' imperative (one that had never been seen by the NER) sandwiched between a lot of narrative (conversational, and hence semantically irrelevant) dialog. We believe that the combination of these two reasons caused the NER to miss that action verb. (Needless-to-say, VERSION2 also missed that action.) As the only difference between the detection portion of VERSION2 and VERSION3 was the addition of the NER to VERSION3, this is extremely strong empirical evidence that, despite the diagnostic test results from Section 2.5.5, the NER was a tremendous help.

Analyzing the recall from the video metric, we see evidence to suggest that the NER output helped there as well. Given how beneficial the NER was above, it is reasonable to suspect that the primary hurdle when judging against the video is the anaphoric description. This is best verified by manual inspection of the predicted tuples, which showed that nearly all of the missed detections (when judged against the video) were due to anaphoric descriptions. This is clearly an area that needs to be worked on, and is discussed in Section 2.7. Regardless, by manual inspection we were able to determine that the recall, for both metrics, is nearly optimized, which gives much credence to the validity of our approach.

The precision is a bit disappointing, although it can be more beneficial to consider the false positive per show statistic of 2.30 rather than the precision percentages themselves. However, as many shows have between 4.70 and 5.07 actions per shows, there can be no denying that the false positive rate is an issue that should be dealt with. In Section 2.4.4 we discussed the certainty measure $c = \mathbf{w} \cdot \mathbf{f}$, which was just a linear combination of four features $\mathbf{f}$. Although the coefficients $\mathbf{w}$ were hand-chosen the certainty measure is a surprisingly good indicator of whether a predicted tuple was actually an action. For instance, nearly all of the 28 predicted tuples with certainty $c < 0.3$ are false positives.

Not only that, but nearly all of the tuples with certainty $c \geq 0.5$ are correct (true) detections. For instance, if all of the 28 tuples with $c < 0.3$ were false detections and we applied a threshold cutoff of 0.3, the precisions would become 80% against the text and 77% against the video. This provides very strong evidence to suggest that with some proper thresholding, and perhaps a more comprehensive certainty calculation, we could significantly reduce the number of false positives detected while not significantly reducing the overall recall. Implementing this was beyond the scope of the workshop however, and is left to future work.

As mentioned above, we used the **70-30:121** classifier in VERSION3. However, we showed throughout Section 2.5 that some classifiers (e.g. **70-30:420** or **70-30:121** in the web-crawling sense) had better recall than **70-30:121**, or just overall better performance (e.g. **100-0:121**). However, the VERSION3 results explain why we decided to use and report **70-30:121** in the system rather than **70-30:420**. While on the web data and in the diagnostic tests we may have gotten higher recall with **70-30:420**, as just discussed, the recall is nearly optimized in VERSION3. On the other hand, the precision for **70-30:420** was much lower than that of **70-30:121**, and our system precision was deemed low enough so that we would not want to do anything to decrease it. And while it would have been nice to use **100-0:121**, since it had both higher recall and precision than **70-30:121** (though in many cases not significantly), the manual evaluation process was very expensive to do and we ran out of time at the end to experiment with it in the system. Further, by using **70-30:121** we have shown what can be done with only 87 well-annotated training files; had we used **100-0:121**, we may have gotten slightly better performance (almost certainly precision-wise), but at the cost of having to say that those numbers required 34 additional well-annotated training files (nearly a 40% increase in the number of required files). The annotation process can be expensive and so we believe that by using **70-30:121** we have presented stronger (and hopefully more generalizable) results.

## 2.7 Future Work

As described in Section 2.2.2, the NER operated on a word granularity: it tagged individual words as being semantically relevant (and how). This though made it difficult to intuitively gauge how the performance of the NER would translate to performance in the overall action-extraction system since in the action-extraction we were focused on was an action-based evaluation, which inherently operated over clauses rather than words. This metric discrepancy will be familiar to anyone involved with speech processing. Many approaches to speech recognition has been focused at the phoneme level, but it is generally considered that a more comprehensive approach is better. Indeed, as seen by systems such as SCARF ([98]), this belief is given empirical supporting evidence. Given the similarities that can be found between the problem of action-related part-of-speech tagging and speech recognition, it would be interesting to adapt a

SCARF-based model rather than just an NER. By doing so we could be abstracting over the words and gathering more semantic information than we currently are, without having to pay the expense associated with typical semantic parsing and understanding.

As mentioned in Section 2.6, one of the biggest problems among all versions was that of anaphoric description. There are multiple ways to try and handle this. The first, and most appealing from a comprehensive A.I. viewpoint, is to more fully incorporate the vision with the language. Specifically, if we are able to predict that there is some action going on (we are just not sure what action exactly), we would ideally like to be able to prompt the vision for certain features (such as common tools, etc.).

However, we may also want to try and find patterns in various action sequences: for instance, is "cut" a more likely action to follow "color," or is "glue?" Indeed the whole group had looked at action transition data from the actual Sprout shows and concluded that due to immense sparsity, the matrix would not be too informative. However, that conclusion was based on the Sprouts data; once we had the new web-data properly annotated, that allowed us to potentially extract action transition probabilities from the web data. With this, we could see if

1. there were any patterns which could be useful in some later stage or iteration of the project, and

2. it would provide as another useful heuristic .

We did not have enough time to fully analyze it in either regard, but with respect to objective 2 we think it seems quite promising.

Two issues that have not been appropriately discussed are those of synonyms and word sense disambiguation. Over the years, it has been determined that WordNet ([28]) is fairly well-suited to that task. Indeed, both [67] and [15] have used WordNet. Using these approaches could be extremely useful, especially when we would want to map all the detected actions to some (much) smaller set; for instance, it would be much too cumbersome and weak if we had to have a classifier/label for each action name even if two were extremely similar (such as "cut" and "tear"). However, there is an additional potential benefit to examining verb similarity: if we used WordNet and corresponding similarity measures, we could potentially greatly decrease our false positive rate, since we might be able to tell (based on the verbs that are most similar to the query verb) whether or not a predicted action was valid. We began to explore WordNet similarity measures, especially with regards to the craft domain, but unfortunately time did not permit a full investigation into this method.

## 2.8 Conclusions

In this chapter, we were presented with the task of automatically generating verb-object-tool tuples from our newly created Sprouts data set. The idea behind these tuples was to have them replace the much more costly human created tuples in the overall action detection and labeling system. We have shown that

we are able to generate such tuples from the text semi-automatically. Since there is some discrepancy between the actions mentioned/described in the text and those enacted on video, two different metrics must be used to comprehensively capture the strengths and weaknesses of our system. When measuring against the text, our best reported results are 99% recall and 69% precision, while when measuring against the video, our best reported results are 92% recall and 65% precision.

Initially, we were not able to achieve those results. We purposefully constructed a very naive baseline, VERSION1, that simply matched four "action-introducing" phrases (which we determined). As this was only to act as an extremely rudimentary baseline, it was not optimized in any way. We then built VERSION2 on top of VERSION1, by using linguistic information, in the form of part-of-speech tags and the dependencies as provided by the Stanford probabilistic parser ([40, 41]), and domain information, in the form of seed action words. This worked very well against both metrics, but the inability of the parser to handle imperatives and the issue of anaphoric description of actions created hurdles which had to be overcome.

As the inability to handle imperatives was not isolated to just the Stanford parser, but was a limitation of many other parsers we experimented with, we decided to adapt a named entity recognizer [32] to act as an action-related part-of-speech tagger. Given the size of the Sprouts data set and the difficulties presented by the narrative structure — and corresponding low semantic density — we decided to train this NER on craft data which we could extract from the web. Initially only 121 instructions out of the 420 total newly obtained craft data were hand-annotated by the authors; these were broken up into training and test sets on a 70/30 basis. We then trained a classifier (**70-30:121**) which was to become our standard and very reliable NER classifier. We experimented with variants of **70-30:121**, including running 'semisupervised' classifiers and simulating a the classifier in a web-crawling set up. These all indicated that the **70-30:121** could be used as an action-related part-of-speech tagger which could be generalized to other domains and uses, such as for cooking shows, home improvement shows, etc.

In our final action extractor (VERSION3) we used the results from running the NER on the Sprouts shows as a way to help compensate for short-comings in the parser. As we ended up with a 99% recall when measured against the text, we have strong empirical evidence to suggest that the NER did what it was intended to do. There is a slight issue of a decrease in precision from VERSION2 to VERSION3, however we are able to use the NER output to construct a "certainty measure" that corresponds extremely well with true positive and false positive detections. There was not enough time to experiment with this certainty measure, but we believe it could be very useful to increasing the precision while keeping the recall at its near-optimal state. The NER also allows us to successfully address the issue of the generation of the seed words used in both VERSION2 and VERSION3. We have therefore shown that we can semi-automatically extract action-related information from the text transcript of an instructional video/show with near-optimal recall.

# Chapter 3

# Extraction of Domain Knowledge

Domain knowledge is one method of using language to aid in object and action recognition in video. Rather than utilizing the video transcript directly, we extract information from the web and other sources of information to develop a semantic background for the results from vision. In this project, this semantic background takes the form of a co-occurrence matrix, which gives a likelihood of a tool and action occurring at the same time in a video. However, domain knowledge to aid in object recognition need not be limited to co-occurrences – we will cover other applications towards the end of this section.

This co-occurrence relationship is known as a "common sense" relationship. Common sense means that the semantic information is understood through common experience, and often applies to daily life. Because they are known to most people, common sense facts rarely appear in text. For example, it is rare to say, "Cut the paper with scissors", because it is understood that scissors are the most common tool for cutting paper. However, we cannot always assume that if we are cutting something, we are using scissors. For example, if one says to "Cut the steak", then common sense knowledge tells us that we should use a knife (and a steak knife, not a butter knife) to cut the steak. Therefore, we must take into account both the object and the domain when determining co-occurrence relationships.

## 3.1 Co-occurrence Matrices

To create the co-occurrence matrices to aid in action and object detection, we experimented with three different sources of domain knowledge – Wikipedia (with WordNet), ConceptNet, and Yahoo search results. Each of these sources have their own strengths and weaknesses, but these sources can be used in conjunction with each other in the global model (which integrates vision and text features). However, since we extract relationships differently from these sources,

the resulting co-occurrence matrices takes two different forms depending on the source – binary for Wikipedia and ConceptNet, or real-valued for Yahoo search results. In our co-occurrence matrix for Wikipedia and ConceptNet, we put a "1" in the intersection of a tool and action if the tool is used to perform the action, and a "0" if this relationship does not apply. In the co-occurrence matrix for Yahoo, we put either a probability of the action and tool co-occurring, or a distance value to indicate the semantic distance between them.

Table 3.1: **Co-occurrence Matrix from Training Data** - *"Writing tool" represents the sum of "pencil", "marker", "pen", and "crayon"*

| | coloring | cutting | drawing | gluing | painting | placing |
|---|---|---|---|---|---|---|
| brush | 0 | 0 | 0 | 1 | 8 | 0 |
| glue | 0 | 0 | 0 | 20 | 0 | 0 |
| scissors | 0 | 38 | 0 | 0 | 0 | 0 |
| writing tool | 12 | 0 | 42 | 0 | 0 | 0 |

## 3.2 Wikipedia Matrix

Wikipedia is an online, user-edited encyclopedia that covers a broad range of topics. We can use the information in Wikipedia to extract the relationships we are looking for. In this method, we also use WordNet, a lexical database for English that categorizes words into semantic categories. It can be used as a dictionary, or to find relationships between related words. Our method for finding action-tool relationships in Wikipedia is rather simple:

1. look up the page corresponding to a particular action,
2. find the nouns, and
3. check WordNet to see which nouns are tools.

### 3.2.1 Method

A few modifications needed to be made to this method to achieve the best results. First, we create the –ing form of the verb in question, because the Wikipedia pages for actions are typically in that form. Second, we can improve the precision at a slight cost of recall by checking nouns only within links and captions in the Wikipedia page. The assumption is that links which are related to the topic will be links in the page, and that links in the page are related to the topic. Captions also provide useful information, because we are ultimately interested in the relationship between textual semantics and vision information. For example, the page for gluing gives a caption, "Nitrocellulose adhesive outside a tube". "Tube" does not appear anywhere else in the document, and it directly correlates to a shape that we would expect to see in video.

### 3.2.2 WordNet

To use WordNet[28, 64] to check our parsed words, we make use of WordNet's relationship properties, which are given for each synset. To find whether a word could be a tool, we retrieve the hypernym paths of each synset containing that word. The hypernym paths provide all of the lexical categories that the synset belongs to – from this list, we can find whether the object belongs to the "implement" or "tool" category. Depending on the domain, we can also extend this to include a broader range of "tools": we would include "utensil" and "chemical" for cooking, for example.

### 3.2.3 Challenges

One drawback of using Wikipedia is that words are not sense-tagged. We must therefore look for nouns that have any possible sense that is a tool, regardless of whether it is used in that context. This approach ensures a high recall, but low precision. While there are some sense disambiguation tools available online, those we evaluated on sample Wikipedia pages were not accurate enough to use. Parsing only the links in the page improved precision at the cost of recall, and solves the sense tagging problem to an extent. However, there are some important tools that are not links in their corresponding action page, such as "pot" in the "boiling" page. Another problem we encountered was the appearance of key words only in articles related to the action's page, but not the action's page itself. For example, "crayon", "marker", and "pencil" were found in the page for "Coloring book", but not in the page for "Coloring". Therefore we expand our search to include a few related pages to ensure that we do not miss tools that may not be in the original article.

### 3.2.4 Results

Our results from Wikipedia were promising in that they matched both intuition and the actual co-occurrences of our training data. In this project, we used a list of tools and actions that we knew would appear in the test data, and restricted the co-occurrence matrix to include only those tools and actions. However, our results could also be used if we did not know the tools (or actions) that would appear in video. For example, we could use online image databases such as image.net or Google Images to retrieve images for a particular tool to train on. An alternative option would be to find physical characteristics of these objects from domain knowledge on the web, and look for these in the video. Work on this method will be presented later in the paper.

## 3.3 ConceptNet

We explored ConceptNet[58] as another knowledge base to extract action-tool representations. ConceptNet is a common-sense semantic network where semantic relationships are added to the database by any web user that has registered

Table 3.2: **Wikipedia Action-Tool Matrix** - *"Writing tool" represents the logical AND of "pencil", "marker", "pen", and "crayon"*

| | coloring | cutting | drawing | gluing | painting | placing |
|---|---|---|---|---|---|---|
| brush | 0 | 0 | 1 | 0 | 1 | 0 |
| glue | 0 | 0 | 0 | 1 | 0 | 0 |
| scissors | 0 | 1 | 0 | 0 | 0 | 0 |
| writing tool | 1 | 0 | 1 | 0 | 0 | 0 |

for the Common Sense Initiative website. This database includes relationships such as PropertyOf, IsA, and UsedFor to define words in terms of semantic relationships with other words.

### 3.3.1 Method

Using ConceptNet, we do not have to do any further processing to find action-tool relationships - we only need to look up the relationships *UsedFor* or *CapableOf* if we want to find the actions a tool is used for. If we want to find the tools used for a particular action, we use the same relationships, but we search for all occurrences of the relationship with the action as the second argument. When creating our co-occurrence matrix, we would put a '1' in the matrix if we found one of these relationships for a given tool and action.

### 3.3.2 Challenges

Since ConceptNet relies on contributions from users of the Common Sense Initiative web applications and not knowledge supplied by a group of experts, there are many gaps in the knowledge base. While ConceptNet does use algorithms for inducing relationships, there were many relationships we expected but did not find.

There were also some errors in the database, such as the tool "saw" being stemmed to form the word "see". Also, while there is some semantic generalization applied to input, often there were details that were parsed from the user input that were not relevant to the semantics of a particular term, and would introduce noise if we did not perform some processing on our results.

### 3.3.3 Results

While the interface of ConceptNet makes finding action-tool relationships simple, we did not use the ConceptNet matrix in the final project because the data was too sparse. In our project, we found only five of the twelve co-occurrences we expected using ConceptNet. Since we obtained better results from Wikipedia, including ConceptNet in our global model could have over-emphasized those co-occurrences that occurred in the ConceptNet results.

## 3.4 Web Matrix

Rather than simply having a binary matrix to say whether an action-tool relationship exists, we would also like to have a continuous measure of how closely related an object and an action are. This helps us determine if one particular object is more closely related to an action than another plausible object. For example, one would think of "crayons" when "coloring", but one can color with colored pencils or markers as well.

These similarity values could be extracted from any sufficiently large and relevant corpus using measures such as point-wise mutual information. Data sparsity is a particular problem for this project since we are working with a specific domain - to find action-tool co-occurrences for arts and crafts shows, we would need corpora specifically relating to arts and crafts. However, since we only need occurrence and co-occurrence data, we can use search engine results from the web to retrieve co-occurrence data from the vast information of the web and thereby avoiding data sparsity problems.

### 3.4.1 Previous Work

Relatedness measures have often been used to find relationships between words. While a relatedness measure can be used to find synonymous words, relatedness (as opposed to similarity) can also be used to find words that share any association, such as automobile and gasoline. Resnik[76] and others have used Wordnet in conjunction with information content from other corpora to determine word similarity. However, Wordnet's relationships are limited to the same part of speech. Therefore, Wordnet similarity measures could not be used to find action-tool relationships.

Web-based relatedness measures have also been explored in previous work. Baroni and Vegnaduzzo [6] used a variant of point-wise mutual information with AltaVista using the "NEAR" operator, which only returns searches that contain the specified words within a certain word distance of each other.

### 3.4.2 Method

To create a matrix with these values, we use a semantic distance measure called the Normalized Google Distance (NGD)[19]. The NGD is a relatedness measure based on the Normalized Compression Distance and modified to use the number of results returned by Google to stand in for the compressor in NCD. In this project, we use Yahoo instead of Google because the Yahoo API is more flexible, and the Google API did not provide the number of search results returned. The implementation remains the same.

To find the semantic distance between two terms $x$ and $y$, we use the Normalized Google Distance equation:

$$NGD(x,y) = \frac{max\{log\, f(x), log\, f(y)\} - log\, f(x,y)}{log(N) - min\{log\, f(x), log\, f(y)\}}$$

where $f(x)$ is the number of search results for query $x$, $f(x, y)$ is the number of search results for query $x\,and\,y$, and $N$ is the total number of pages indexed by the search engine. This returns a value between 0 and infinity (because of the details of how Google calculates the number of results), but most values returned are between 0 and 1. The lower the number, the more related two queries are, with two identical queries having a distance of 0. We calculate the NGD for each action-tool pair and enter it into a matrix to form our co-occurrence matrix.

### 3.4.3 Modifications

**Verb Forms** An issue with using the web for a distance measure is that many verbs have a noun form that is identical to the verb form. We achieve results more correlated with our expectations by using the -ing form of a verb instead of the infinitive form. This modification restricts our results to the action we are interested in and avoids noun forms in our search.

**Domain-Specific Comparison** Action-tool relationships can differ according to a particular domain. For example, in arts and crafts, one associates cutting with scissors. However, in cooking, one would associate cutting with a knife. We would like to capture these domain-specific relationships to improve the accuracy of our reported similarity. To restrict to a particular domain using the NGD, we append the domain to each of our queries. For example, to determine $f('scissors')$, we perform the query <scissors "arts and crafts">. Likewise, to determine $f('scissors','cutting')$, we perform the query <scissors cutting "arts and crafts">.

**Domain Scaling** We found that simply adding the domain to the search query did not completely disambiguate domain-specific relationships. Even if we specify the domain as "arts and crafts", we obtain a smaller distance between "knife" and "cutting" than we do between "scissors" and "cutting". We further enforce domain specificity by multiplying our distance by a scaling term, which is the distance between the object and the domain. Thus, if a particular object is not associated with the domain, it will be given a larger distance. The equation is then as follows:

$$SNGD(x, y) = NGD(x, y) * NGD(x, domain)$$

where $x$ is the object and $y$ is the action.

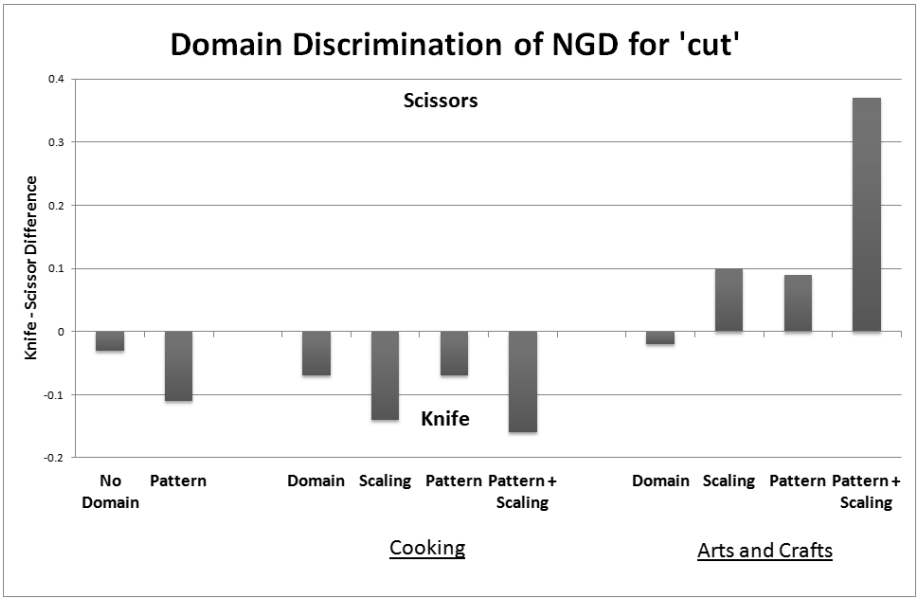
**Pattern Matching** Our implementation so far has been searching for a co-occurrence of two words anywhere in an entire web page. However, given the variety of content on a typical web page, the context of these two words may be different, and their co-occurrence may be entirely incidental. We would like to discard these matches in our search query, and work under the assumption that two words are related only if they occur within a certain word distance within each other. We achieve this by using the pattern matching feature that

is provided by the Yahoo (and other search engines) API. This pattern matching feature allows us to perform similar queries to those we could make with AltaVista's "NEAR" operator. If we change our query to < "scissors * cutting" OR "cutting * scissors" "arts and crafts" >, the search will only return pages where "scissors" and "cutting" appear within two words of each other. This preferable to searching for words that are adjacent to each other, given that there are often prepositions or other words in between a tool and an action. One can also increase the distance up to within 5 words of each other by adding additional asterisks. To search for words within three words of each other, the query would be < "scissors * cutting" OR "cutting * scissors" OR "scissors * * cutting" OR "cutting * * scissors" "arts and crafts" >. We found that a distance of two words provided results that most coincided with our expectation of co-occurrence.

### 3.4.4 Results

We tested the domain scaling modification to see how well it performed domain discrimination when objects came from different domains. We then compared the different domain discrimination techniques on our scissors and knife example mentioned previously. We would like to show that a knife is more related to cutting than scissors in the cooking domain, but scissors are more related to cutting in the arts and crafts domain. We find that domain scaling successfully enforces this relationship, but pattern matching also aids in domain discrimination to a lesser extent. The results are shown in Figure 1. One drawback of domain scaling is that it can introduce noise to the distance values when objects are all of the same domain, and since that is true of the objects we will find in our video, we did not use it in our final project.

Figure 3.1: Domain Discrimination Results




Our results for the Web matrix correlate with both our expectations and the co-occurrences of the training data. Table 3 shows the final matrix that was used in the global model. In this matrix, the action-tool pairs that we expect to see in video have the lowest distance values. In this case, values below 2.0 indicate that an action-tool relationship exists, and this pattern provides the co-occurrence information that can be used in the global model.

Table 3.3: **Web Matrix** - *Distance values from the Normalized Google Distance with a specified domain and pattern matching. Lower values indicate two terms are more related. INF values indicate little or no co-occurrence, and were smoothed to a high distance value in the model. "Writing tool" represents the average of "pencil", "marker", "pen", and "crayon".*

| | coloring | cutting | drawing | gluing | painting | placing |
|---|---|---|---|---|---|---|
| brush | 2.51 | 2.11 | 2.40 | INF | 1.85 | INF |
| glue | 2.51 | 2.51 | 2.51 | 1.2 | 2.44 | INF |
| scissors | 2.47 | 1.76 | 2.36 | INF | 2.68 | INF |
| writing tool | 2.12 | 3.51 | 1.72 | INF | 2.08 | INF |

### 3.4.5 Discussion

Our modified NGD algorithm can have other uses aside from constructing a relatedness matrix. For example, we may have a large number of results from

Wikipedia mining, but some may not be related to the particular domain we are interested in. We can use our distance algorithm to prune results that are not specific to the domain by using domain scaling and pattern matching.

# Part II

# Computer Vision

# Chapter 4

# Action Recognition

## 4.1 Introduction

The analysis of actions and activities from videos is a very active research topic in Computer Vision due in part to several promising applications such as surveillance, elderly care, human-machine interfaces, video-indexing based on actions, etc. However even after almost two decades of research, the problem of automatic action recognition remains mostly unsolved. There are numerous challenges for the development of robust and general action recognition algorithms: scene and self occlusions, environmental affects such as lighting, in-door vs outdoor, clothing, carry-on accessories, technical affects such as video size, sampling rate, camera motion, as well as myriad other factors such as multiple actions/humans, human/object interactions, different semantic interpretation of the same scene, etc.

Most of the work in the area of automatic action recognition considers whole body actions such as walking, running, jumping, etc. This is also the scenario that is most common in surveillance applications. Over the past years, several benchmark datasets have emerged that contain relatively modest variations in camera motion, scale, clothes of the actors, etc., for the development of robust action recognition algorithms. State-of-the-art methods also show promising recognition rates on these datasets. Recently, there has been a shift towards recognition of not just whole body actions but also human interactions with other humans as well as objects. Moreover, for elderly care applications, it is important to recognize activities of daily living such as cooking and cleaning, etc. The activities of interest here are manipulation tasks such as cutting food, gluing materials or painting which could have very different appearance depending on the camera location. In this section, we consider the automatic recognition of hand-crafts such as cutting, coloring, gluing, painting, etc. We will show that state-of-the-art approaches that only consider spatial-temporal patterns of motion and intensity of the moving objects do not perform well on manipulation tasks such as ours. However, these approaches still provide meth-

ods for generating robust features for action representation which will be used later.

### 4.1.1 Prior Work

There is a long history of human motion analysis in computer vision. The surveys by Gavrila ([33]), Aggarwal et al. ([2]), and by Moeslund et al., ([65, 66]), provide a broad overview of over three hundred papers and numerous approaches for analyzing human motion in videos, including human motion capture, tracking, segmentation and recognition. Most of the work in activity recognition can be divided into two classes: 1) collections of local models and 2) global models. Local models compute a collection of spatio-temporal interest points such as the ones defined in [24, 48, 90] and compute a descriptor based on intensity, optical flow and their gradients in a spatio-temporal cuboid centered at each interest point ([24, 50, 90]). The action in a particular video is then modeled as a distribution over a dictionary of codewords learned from a large collection of these spatio-temporal descriptors extracted from a database of human actions. Wang et al. ([88]) provide a comparison of the recognition performance of various 3D interest point detectors and their corresponding descriptors on standard datasets such as the Weizmann dataset ([34]), KTH dataset ([78]) and the Hollywood action datasets ([50, 62]). Another local approach proposed by İkizler et. al. ([36]) learns local limb motion models from labeled motion capture data to query more complicated actions as a composition of these local models.

On the other hand, global models for human actions compute statistics of motion and intensity over the whole frame or an extracted human skeleton or silhouette. Efros et al. ([26]), modeled human actions at a time instant by histograms of optical flow. These histograms were then used to match the motion of a player in a soccer match to that of a subject in a control video. Tran et al. in ([83]) present an optical flow and shape based approach that uses separate histograms for the horizontal and vertical components of the optical flow as well as the silhouette of the person as a motion descriptor. Gorelick et al. ([34]) and Yilmaz et al. ([94]) represent human activities by 3-D space-time shapes. Classification is performed by comparing geometric properties of these shapes against training data. Another class of global models specifically models the temporal dynamics of human activities. In general, the recognition pipeline is composed of 1) finding features in every frame, 2) modeling the temporal evolution of these features with a dynamical system, 3) using a similarity criteria, e.g. distances or kernels between dynamical systems, to train classifiers, and 4) using them on novel video sequences. Bissacco et al. used joint-angle as well as joint trajectories from motion-capture data ([10]) and features extracted from silhouettes in [11] to represent the action profiles. Ali et al. in ([3]) used joint trajectories to extract invariant features to model the non-linear dynamics of human activities. Chaudhry et al. in [16] use non linear dynamical systems to model the temporal evolution of optical flow histograms.

As mentioned earlier, all the above approaches model full body human actions such as running, walking and jumping etc. With the recent interest in

manipulation tasks, datasets such as the CMU Quality of Life Grand Challenge Kitchen Dataset ([1]) and the University of Rochester Activity of Daily Living Dataset (URADL) ([63]) have been released. Benchmark results such as those presented in [63] show that state-of-the-art approaches that use spatio-temporal interest points do not scale well. Instead, an inference approach using velocity features extracted by tracking points across frames performs much better. Recently, another tracking based approach was proposed by Raptis et al. in [74] where spatio-temporal interest points were extracted and tracked. At each tracked location a descriptor that averaged the intensity gradients or the flow was computed. A bag-of-words like approach using these features was shown to get promising results on both the KTH dataset as well as the URADL dataset.

Finally, there has been some work at recognizing actions in individual images and frames. Most relevant is the recent work by Yao et al. ([93]) where object detectors and human pose detectors are jointly used to classify the object and pose (action performed) of the human.

### 4.1.2 Contributions

In this chapter we will evaluate some of the most common action recognition approaches on the Sprout TV Crafts dataset. In section 4.2, we describe three supervised approaches for shot-level action recognition. In section 4.3, we describe a simple Multiple Instance Learning based approach for automatically learning the labels of the individual shots in an episode by using episode-level labels only.

## 4.2 Supervised approaches

### 4.2.1 Global Histograms of Oriented Optical Flow

The optical flow of a scene describes the motion characteristics of moving objects in a scene as well as the motion of the camera. If the camera is stationary, the optical flow is only the result of object motion. Moreover, if there is only one person in the scene, the optical flow at each frame is characteristic of the state of motion at that time. To model the temporal evolution of the motion of optical flow, Chaudhry et al. proposed a system theoretic approach in [16] where the variation in the optical flow signature caused by the human action is modeled as a Non-Linear Dynamical System (NLDS). Given a video, the optical flow is computed for each pair of consecutive frames. The flow in each frame is then quantized into $B$ bins according to the direction of the optical flow vector. More specifically, for the optical flow vector $(u, v)$, at pixel location $(x, y)$, and having magnitude $m = \sqrt{u^2 + v^2}$ and angle, $\theta = \arctan \frac{y}{x}$ contributes to bin $b$ where,

$$-\frac{\pi}{2} + \pi\frac{b-1}{B} \leq \theta < -\frac{\pi}{2} + \pi\frac{b}{2} \tag{4.1}$$

The contribution of each optical flow vector is equal to its magnitude $m$. The resulting histogram is then normalized to one to make it invariant to the scale

Table 4.1: 1-NN Classification using global laterally invariant HOOF (Metric labels are: M = Martin, BC = Binet-Cauchy, BC-MSV = Binet-Cauchy Maximum Singular Value, KL = KL Divergence, Means = Difference of temporal means. Hybrid metrics use a linear combination of the corresponding metric and the difference of temporal means. See [17] for more details)

| Metric | Regular | Hybrid |
|---|---|---|
| M | 27.17 | 26.09 |
| BC | **28.26** | 26.09 |
| MC-MSV | **28.26** | **28.26** |
| KL | 13.04 | 22.83 |
| Means | 22.83 | |

of the person in the scene. The quantization of the 2-D space according to Eq. (4.1) makes the histogram feature invariant to lateral motion of the person. This is especially useful when we do not want to distinguish between, for example, running left and running right. For each video, a time-series of these Histograms of Oriented Optical Flow (HOOF) is extracted. Each time-series is then modeled by an NLDS and metrics defined on the this space of NLDS are used to compare HOOF time-series. Any classification algorithm such as k-Nearest Neighbors (k-NN) or Support Vector Machine (SVM) can then be used to classify actions in novel video sequences based on learnt models from a training set of interesting actions. For more details, we refer the reader to [16].

For our first set of experiments, we focus on the manually annotated shots in the original episodes. Of all the 30 annotated actions, there are 11 actions that have more than 5 annotated shots per action with a total of shots. We use 50% of the shots for each action as training and the remaining for testing. To compute the optical flow for each shot, we use the GPU-based TV-L1 norm optical flow method in .We divide the 2D +ve x-axis plane in 64 equal parts according to the angle to the horizontal. This gives us 64 bins for computing the HOOF features. Once a time-series of HOOF features has been computed for each shot, we use the kernel $k(h_i h_j) = \sum_k \sqrt{h_i(k) h_j(k)}$, to compute the system parameters of the NLDS that generates this time series. We use a system order of 10. In this way, the system parameters for all the training and testing shots are computed. To find the action label of a test sequence, we compute the distance between the parameters of the test shot and all training shots and assign the label of the training system that is the closest to the test system. Table 4.1 shows the results for 1-NN classification using several dynamical systems metrics.

As we can see, the best recognition rate achieved is 28.26% which is only marginally better than the recognition rate achieved by simply assigning the label of the most frequent class in the training set to all test samples, 22.83%. Figures 4.1 and 4.2 show the confusion matrices for the regular and hybrid Binet-Cauchy maximum singular value kernel when used to compute the distance between two NLDS in the 1-NN classification scheme. As we can see, the performance is very poor and the classes that have larger number of sequences

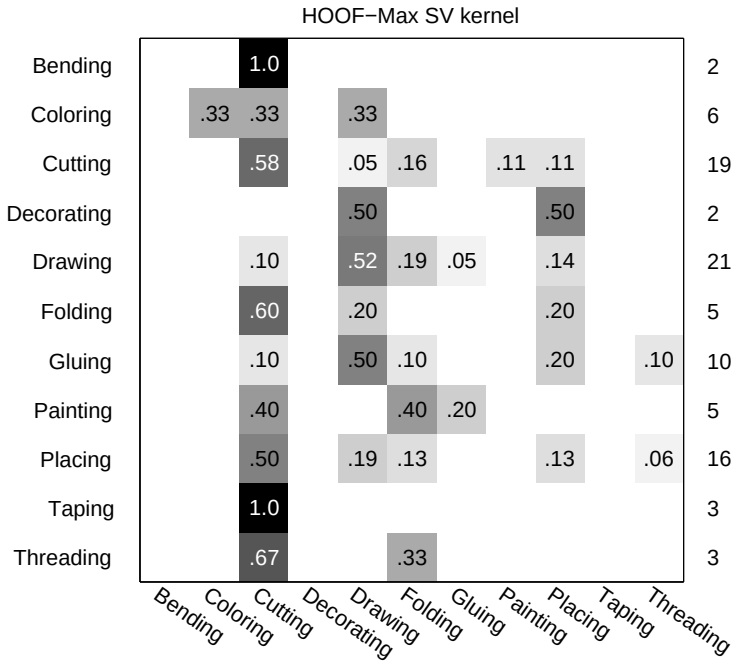



Figure 4.1: Confusion matrix for 1-NN classification using the regular Max SV kernel on the test set from the manually annotated shots. The right most number indicates the number of test samples for each class.

seem to overshadow the ones that have a smaller number of sequences. For example, most of the sequences are classified as Cutting or Placing which are the most dominant classes.

When using a version of HOOF that is not laterally invariant, by dividing the whole 2D space into $B$ bins instead of just the +ve x-axis, we get get slightly better results as shown in Table 4.2 with the best result achieved with the Martin distance. Figure 4.3 shows the corresponding confusion matrix and as we can see that qualitatively there is not much of a difference from the previous confusion matrices.

One of the main reasons that the HOOF based global approach is not working well is that the videos contain a number of artifacts such as camera motion, zooming and fading. These artifacts cause spurious optical flow patterns that are not a result of the motion of the objects of interest. The dynamical systems estimated thus, are not entirely representative of the action but also try to model these spurious optical flow signatures. Since the artifacts are unpredictable and vary in frequency and form across the videos, it is very difficult to remove them from the original video. The global HOOF based approach hence tends to perform poorly. We also see the effects of insufficient training data for some

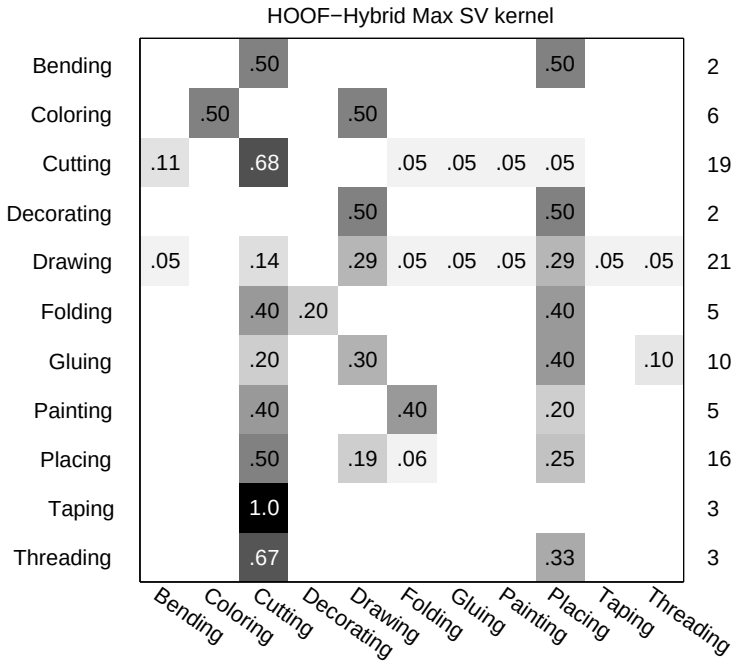



Figure 4.2: Confusion matrix for 1-NN classification using the Hybrid Max SV kernel on the test set from the manually annotated shots. The right most number indicates the number of test samples for each class.

of the classes. Table 4.3 shows the recognition results when we only focus on the three most frequent classes: Cutting, Drawing and Placing. Similarly Table 4.4 shows the recognition results when we only focus on the two most frequent classes: Cutting and Drawing. As we have a smaller number of classes and a relatively balanced number of sequences per class, the recognition results are better. However because of the camera artifacts, the results are poor, and as we will see later, local approaches tend to perform much better.

### 4.2.2 Local Spatial-Temporal Interest Points

Global models for action recognition tend to perform poorly whenever there are camera or scene artifacts such as moving cameras, self occlusions, etc. For object recognition in images, local features such as SIFT [59] have proven to be very useful. These features have a small spatial extent and a complete object is represented as a sum of these local features as a frequency distribution over a set of automatically chosen representative features. Laptev et al. in [48] showed that similar features can be constructed for videos by extracting corners in space-time instead of just image space. In a similar fashion, Dollar et al. in [24] presented an alternative method for extracting candidate space-time points

Table 4.2: 1-NN Classification using global laterally invariant HOOF (Metric labels are: M = Martin, BC = Binet-Cauchy, BC-MSV = Binet-Cauchy Maximum Singular Value, KL = KL Divergence, Means = Difference of temporal means. Hybrid metrics use a linear combination of the corresponding metric and the difference of temporal means. See [17] for more details)

| Metric | Regular | Hybrid |
|---|---|---|
| M | **33.70** | 32.61 |
| BC | 23.91 | 22.83 |
| MC-MSV | 27.17 | 26.09 |
| KL | 17.39 | 19.57 |
| Means | 17.39 | |

Table 4.3: 1-NN Classification for 3-classes using global laterally invariant HOOF (Metric labels are: M = Martin, BC = Binet-Cauchy, BC-MSV = Binet-Cauchy Maximum Singular Value, KL = KL Divergence, Means = Difference of temporal means. Hybrid metrics use a linear combination of the corresponding metric and the difference of temporal means. See [17] for more details)

| Metric | Regular | Hybrid |
|---|---|---|
| M | **55** | 48 |
| BC | 41 | 48 |
| MC-MSV | 46 | 52 |
| KL | 38 | 43 |
| Means | 43 | |

Table 4.4: 1-NN Classification for 2 classes using global laterally invariant HOOF (Metric labels are: M = Martin, BC = Binet-Cauchy, BC-MSV = Binet-Cauchy Maximum Singular Value, KL = KL Divergence, Means = Difference of temporal means. Hybrid metrics use a linear combination of the corresponding metric and the difference of temporal means. See [17] for more details)

| Metric | Regular | Hybrid |
|---|---|---|
| M | **75** | 70 |
| BC | 60 | 68 |
| MC-MSV | 73 | 78 |
| KL | 65 | 65 |
| Means | 70 | |

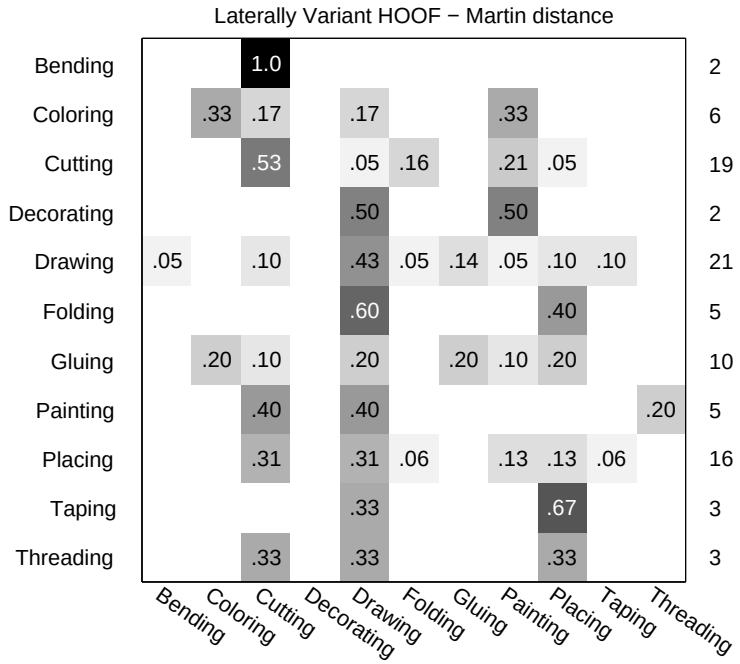



Figure 4.3: Confusion matrix for 1-NN classification using the Martin distance with Laterally Variant HOOF on the test set from the manually annotated shots. The right most number indicates the number of test samples for each class.

by filtering the video volume with spatial Gaussian and temporal Gabor filters. In general these Space-Time Interest Points (STIP) can be extracted at several spatial and temporal scales corresponding to the scales of the various filters used. Once these STIP have been extracted, a feature is extracted at each STIP that describes the spatial and temporal characteristics of intensity and flow in a neighborhood of that point. Commonly used features are intensity, optical flow and their derivatives. In particular Laptev et al. in [50] introduced the Histogram of Gradients (HOG) and Histogram of Flow (HOF) features which have shown promising results on several human action datasets. Once local features have been extracted, a set of representative features or codewords is extracted by clustering all the features extracted from a training set. The cluster centers represent areas of high density in the feature space and a shot containing several features can thus be represented as a frequency distribution over these codewords. With each shot thus represented as a histogram, we can use any histogram metric such as the $\chi^2$ distance to compare different action shots using any classifier such as 1-NN or SVM.

To parallel the first experiment in the previous section, we use publicly

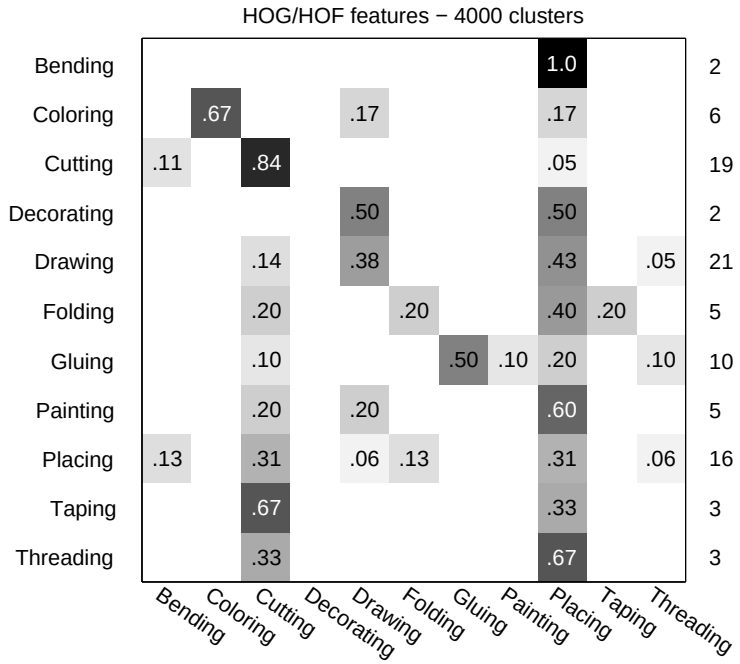



Figure 4.4: 1-NN classification, recognition rate 42.39%

available code[1] provided by Laptev et al. to compute STIP and HOG+HOF features from all the manually annotated shots. This extracts STIPs at spatial scales of $\sigma^2 = \{4, 8, 16, 32, 64, 128\}$ and temporal scales, $\tau^2 = \{2, 4\}$. With the key point extracted, an $18\sigma \times 18\sigma \times 8\tau$ cuboid is considered with the key point at the center. This cuboid volume is further divided into $3 \times 3 \times 2$ parts and for each part a 4-dimensional Histogram of Gradients (HOG) and a 5-dimensional Histogram of Flow (HOF) is extracted to get a $18 \times (4 + 5) = 162$ dimensional feature. From the training dataset consisting of 50% of all annotated actions, all the features are extracted and clustered in 100 and 4000 clusters (codewords). For each shot, the term frequency of each codeword is computed by finding the number of features in each shot that are closest to that codeword. After normalization, we get a histogram of term frequency for each shot. Using the $\chi^2$distance and 1-NN classification, the recognition results on the remaining 50% data used as test data are shown in the confusion matrices in Figures 4.4 and 4.5.

As we can see the recognition rate is better than the global HOOF approach. Moreover, we see a better diagonal structure in the confusion matrix. However, the problem of varying number of samples per class manifests itself in this as well. As we can see a number of smaller classes are classified as Cutting or

[1] http://www.irisa.fr/vista/Equipe/People/Laptev/download/stip-1.0-winlinux.zip

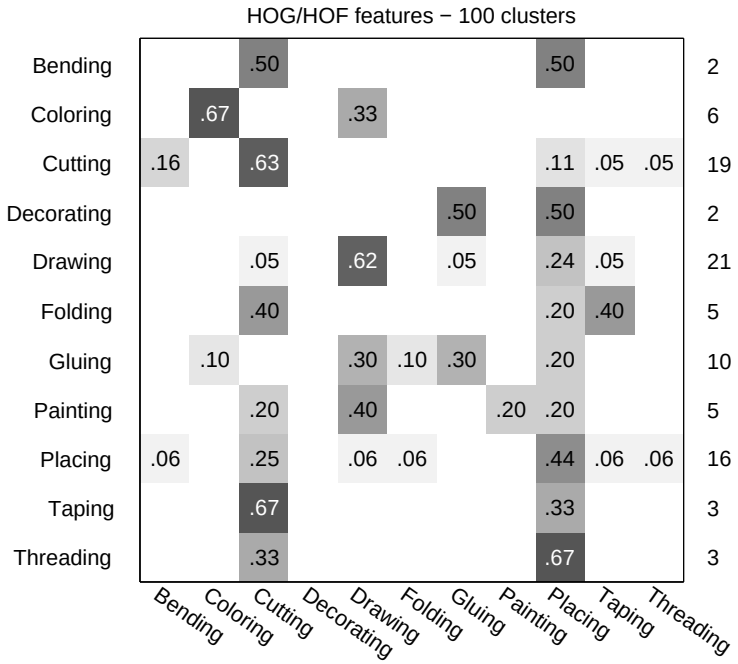
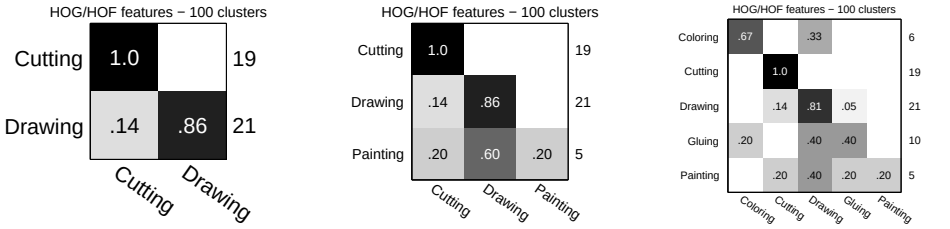



Figure 4.5: 1-NN classification, recognition rate 43.48%

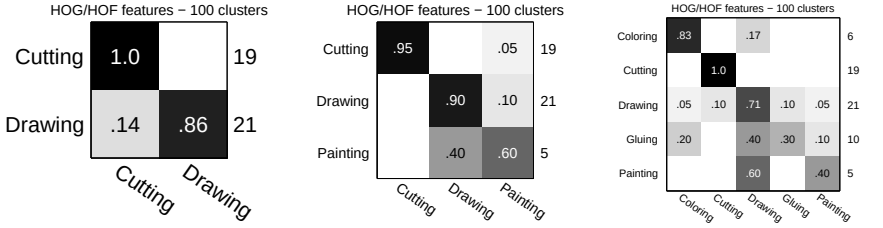
Placing, the classes with larger number of sequences. Figure 4.6 shows the classification results for the 2-, 3-, and 5- class sub problems. As we can see the sequences with smaller number of classes tend to be confused with the more frequent classes. The small number of training samples creates a challenge to learn a good 1-NN classifier. Even with balancing the number of training samples by choosing the same number of samples per class, the results are no better than choosing all as can be seen in Figure 4.7. Moreover, although not shown here, the results depend heavily on the initial sample set used in training.

For the second part of our experiments, we focus our attention on automatically extracted shots using a shot-segmentation algorithm. Moreover, we only consider the 5 most common actions: Coloring, Cutting, Drawing, Gluing and Painting. As part of training, we choose 13 out of the 27 full episodes and the remaining 14 are used for testing. The labels of the manual annotations are transferred to the automatically extracted shots. The label of a manually annotated shot is transferred to an automatically segmented shot if at least half of the latter overlaps with the former. We observe that almost all automatically segmented shots overlap nicely with the manually annotated ones. Action shots from other classes are given the label "Other" or "Uninteresting". Zoomed-out shots can be detected using a context-detection algorithm and with almost 100% accuracy removed for further analysis. All zoomed-in shots with the correspond-




(a) Recognition rate 92.5% (b) Recognition rate 84.44% (c) Recognition rate 73.77%

Figure 4.6: Variation of recognition rate with number of classes




(a) Recognition rate 92.5% (b) Recognition rate 88.89% (c) Recognition rate 73.77%

Figure 4.7: Variation of recognition rate with number of classes with equal training samples for each class

ing transferred labels are used for training and testing by using the same feature extraction pipeline as above.

Figure 4.8 shows the confusion matrix for the above mentioned scenario. The recognition rate is 50%, whereas labeling every sample with the label of the most frequent class - Uninteresting - is 47%. The large number of training and testing samples for the Uninteresting class again biases the results towards high. However, the average class recognition rate, which is the average of the diagonal values of the confusion matrix is 43.33% which is much better than the corresponding chance level of 17%.

Since the STIP HOG+HOF features and the corresponding bag-of-features approach gives the best results on this dataset, we will use the resulting frequency histogram per shot as the *action feature* to be used later when combining with other features such as tool and hand detectors as well as co-occurrence statistics from language.

### 4.2.3 Local Histograms of Oriented Optical Flow

As a compromise between the global and local feature, we propose to extract STIP key points as in the previous section, but as a feature, we compute the HOOF NLDS as in [16] for each cuboid. For the clustering stage to compute

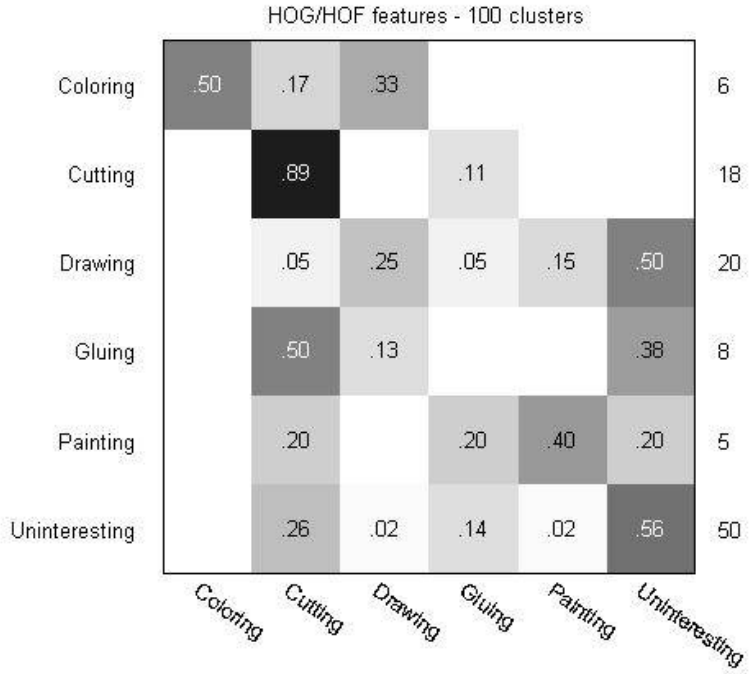



Figure 4.8: Recognition rate 50%, average class-level recognition rate 43.33%

codewords, we need to cluster NLDS. This is not trivial as the space of NLDS is not Euclidean and a simple k-means procedure on vectorized parameters of the NLDS will not give the correct answer. We choose to use the approach described in [75], which uses MDS on the all-pair dynamical systems based distance matrix to embed all the NLDS in a lower-dimensional Euclidean space and perform clustering on that space. As cluster representatives in the NLDS space, the data-point closest to each cluster center in the lower-dimensional Euclidean space is chosen. When computing term frequency histograms, the assignment of a feature to a codeword is done by finding the codeword that is the closest to the feature in terms of the dynamical systems based distance.

We only perform one experiment on the manually annotated shots with the 50% training and 50% test split as before. To find keyword candidates in training data, we use the Martin distance to find all pairwise NLDS distances, perform MDS on these to get 90 dimensional Euclidean vectors. K-means is used to cluster these vectors in 100 clusters as before to compare to HOG/HOF. Term-Frequency histograms are computed for the training videos. For cuboids in test videos, the Martin distance is used to compute keyword memberships before computing TF histograms. The remaining classification procedure is the same as the usual bag-of-features approach.

Unfortunately, computing the all-pair distance matrix of all local HOOF

features is not possible because of the sheer number of local HOOF systems in the training set ~ 800k. To keep the computation tractable, we sample 1000 candidate points and cluster these into 100 clusters to find the keywords. On the 2-class sub-problem, we get a recognition rate of 67.5%.

Since this approach does not perform as well as the local STIP HOG+HOF approach, we did not perform any more experiments and as described earlier, the *action feature* to be used from now on per shot will be the frequency histogram over STIP HOG+HOF feature codewords as described in the previous section.

## 4.3 Unsupervised approach

Labeling all the shots manually even for training purposes is an expensive task as it could possibly take hundreds of man-hours to annotate shot boundaries and all corresponding actions in a set of video episodes. Since our dataset provides a transcript for each video, we could extract possible episode level labels from the transcript by extracting action verbs from the transcript. In this section, we develop an approach that takes episode-level labels and automatically computes all the shot-level labels for each episode. This problem naturally falls in a Multiple Instance Level (MIL) framework where we can create *positive bags* for each action label as the episodes that contain at least one instance of that action and *negative bags* that contain no instances of that action. Figure 4.9 provides an illustration of this.

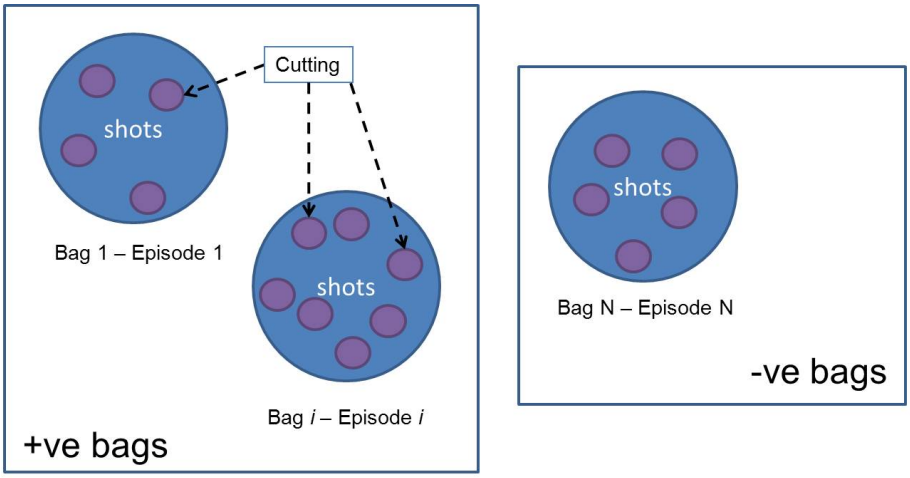



Figure 4.9: +ve and -ve bags formed from episodes of our dataset

In the following, we will briefly describe the Diverse Density algorithm [61] and then apply a multi-class version to our problem.

Table 4.5: 1-vs-all binary classification for each action

| Accuracy (%) | Bag Level | Instance Level |
|---|---|---|
| Coloring | 71 | 94 |
| Cutting | 50 | 20 |
| Drawing | 57 | 22 |
| Gluing | 50 | 10 |
| Painting | 86 | 95 |

### 4.3.1 Diverse Density algorithm for MIL

The Diverse Density [61] algorithm aims at finding areas in feature space that have a high density of positive instances and a low density of negative instances of the class of interest. Denoting positive bags as $B_i^+$, the $j$th instance in that bag as $B_{ij}^+$ and the negative bags as $B_i^-$, we want to find $x$, the point in feature space that maximizes, $P(x = t|B_1^+, B_2^+, \ldots, B_n^+, B_1^-, B_2^-, \ldots, B_m^-)$ where $t$ is assumed to be the point in space that represents the concept. Using Bayes' rule and independence assumptions on the bags, this is equivalent to,

$$\text{argmax}_x \prod_i P(x = t|B_i^+) \prod_i P(x = t|B_i^-). \tag{4.2}$$

Using $P(x = t|B_i^+) = P(x = t|B_{i1}^+, B_{i2}^+, \ldots) = 1 - \prod_j (1 - P(x = t|B_{ij}^+))$, and $P(x = t|B_i^-) = \prod_j (1 - P(x = t|B_{ij}^-))$, and the probability definition, $P(x = t|B_{ij}) = \exp(-\|B_{ij} - x\|^2)$, the above problem is solved using gradient ascent. Since the optimization function is non-convex, multiple initializations are used to find the optimum $x$. Moreover, each dimension of the feature can be weighted by a scaling factor $s_k$ in the distance definition, $\|B_{ij} - x\| = \sum_k s_k^2 (B_{ijk} - x_k)$ and the optimum $s$ found during the gradient ascent procedure. Once the optimum $x$ has been found, a new instance is labeled as belonging to the class if its probability is above a certain threshold using the exponential probability definition just described. For more details, please refer to [61]. We used the code provided in the Multiple Instance Learning Library (MILL) Toolkit[2] in our experiments. It provides a number of MIL methods, however we chose to use Diverse Density following our arguments above.

Following our previous training-testing breakdown, we use 13 episodes for training and the remaining episodes for testing. At first we consider each class separately and use the default binary MIL classifier provided with the toolkit.

Table 4.5 shows the result of MIL for the 1-vs-all scenario. In each case, the episode bag is labeled as +ve if it contains at least one instance of the class of interest and -ve otherwise. The default threshold of 0.5 was used in these experiments. Bag Level accuracy illustrates the percentage of times a bag in the test data was correctly labeled as +ve or -ve depending on the occurrence of instances of the class of interest. Instance Level accuracy details

[2] http://www.cs.cmu.edu/~juny/MILL

Table 4.6: Automatic labeling of the whole dataset

| Action | # +ve bags / 27 | Accuracy (%) |
|---|---|---|
| Coloring | 6 | 94 |
| Cutting | 17 | 80 |
| Drawing | 18 | 77 |
| Gluing | 13 | 89 |
| Painting | 4 | 94 |

the percentage of action shots that were labeled correctly as +ve and -ve in all the test episodes. As this is a binary classifier, the results are not comparable to the results shown in the previous section. Moreover the accuracy is high even when all the instances are classified as -ve due to the large amount of -ve instances in the data.

Table 4.6 displays the results when we considered the whole dataset - all 27 videos - and assumed that all the bag-level labels and a few positive instance level labels were known a-priori. The +ve instances are used as starting points in the gradient ascent procedure. Eventually we are able to compute the labels for all the points in the dataset using a 1-vs-all classification scheme for each action. The high accuracies are promising, however one reason for the high rate is the large number of -ve labels for each action. Consider that of all the 186 annotated actions, the highest occurring class, Drawing, has a total of 42 sequences, whereas the lowest occurring class, Painting, has a total of 11 sequences. Hence if we label all the sequences as -ve, the default accuracies are at least 77.4% and at most 94%. Since the accuracy achieved lie in this range, our results are unfortunately inconclusive.

Finally, we tested our method on the automatically extracted zoomed-in shots. We implemented an all-vs-all scheme that encapsulates the binary classifier provided by the MILL toolkit to perform multi-class classification. Moreover, the threshold was learnt using a 5-fold cross-validation scheme on the training set for each binary classifier (there being ${}^6C_2$ of these for each class: Coloring, Cutting, Drawing, Gluing, Painting and Uninteresting). Again, 13 videos were chosen as training and 14 for testing. The instance level accuracy achieved was **46.98%** but unfortunately all the instances were labeled as belonging to the Uninteresting class and the high frequency of instances of that class led to this high recognition rate.

Overall our MIL experiments have been inconclusive, however we observe that this has mostly been due to the small amount of data. MIL algorithms generally require large amounts of training data to be successful. Given the original motivation of only having to label episodes and only a few instances, developing a method that automatically extracts all instance level labels in a training set and generalizes to test data remains a promising research path.

## 4.4 Summary and Future Directions

From the above discussion, we have seen that state-of-the-art action recognition approaches do not scale well to a realistic scenario with limited amounts of training data in a complicated domain. Moreover, the results achieved by applying global approaches designed for human actions to manipulation tasks in real world videos, containing many camera artifacts, are not promising. Local feature based approaches perform the best but still the recognition rates are not very good. There is clearly a need for modeling the Uninteresting class differently from the interesting classes. Context and Domain knowledge will need to be added to the scheme to make it successful.

Overall, the STIP HOG+HOF feature pipeline with the bag-of-words approach has provided a good action representation that will be combined with textual, object and hand features later to achieve better results than the ones shown here. As future work, the idea of finding the best feature for action representation is still an open research question. We are looking into creating other combinations of flow and texture feature distributions over time, as well as using motion trajectories. Finally, we are looking at other MIL algorithms that might not need such a large amount of training data as Diverse Density and ideally, we would like to train using labels extracted using MIL and action-names from textual analysis instead of ground-truth annotations to make the overall recognition pipeline as unsupervised as possible.

# Chapter 5

# Object Detection

## 5.1 Introduction

Object detection is one of the fundamental problems in computer vision. It deals with detecting instances of particular objects (such as humans, cars, buildings) in images and videos. It is a difficult problem because the appearance of objects can vary due to changes in illumination, viewpoint and also due to deformations. An image is composed of many objects, many of them occluding each other and appearing in different poses. The possible variations(like deformations, shape and appearance) present within a class makes it unlikely that we can simply perform an exhaustive search based on a database of class examples.
In the problem that we explore, the ability to predict the absence or presence of a particular class of objects can provide us with strong cues to determine possible actions being performed by the subject. For example, a high confidence for presence of a tool like scissor or knife can indicate the possibility of a cutting action being performed in the scene. We present an approach which use state of the art object detectors and extend them to providing us with cues to the presence of particular objects in a sequence of frames.

## 5.2 Related Work

The "bag of words" model ([21, 97]) quantizes local image descriptors into visual words which can then be used to describe different object classes. Local descriptors like [60] are first extracted from images. These are then clustered into visual words using approaches like k-means ([81]) or hierarchical clustering ([68]). The system then computes histogram of visual words using the computed visual vocabulary and can use approaches like Naive Bayes or Support Vector Machines for classification. However these models do not try to learn relative locations of the visual words. Detecting objects using shape features has also been explored by some. [7] measure similarity between shapes and exploit it for recognition. They compute a shape context descriptor for a set of points on

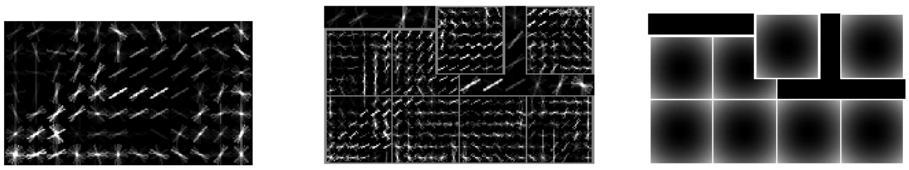
the contours of a shape. The shape context at a reference point captures the distribution of the remaining points on the contour relative to it and offers a globally discriminative characterization.
Sliding window classifiers are well suited for rigid objects and have been used for detection of faces, cars and pedestrians in [71, 87]. The approach involves scanning the image with fixed templates and applying a classifier to the sub-image defined by the window. The process may be repeated at multiple scales so that objects can be detected at any size. However they fail to take into account contextual cues and have difficulty in detecting classes with large intra-class variation. Parts-based models ([30, 29]) represent objects as set of parts. They model the relative location between parts and learn the appearance of individual parts. The advantage with such models is that local features for the individual parts provide information about the appearance while loose geometric constraints can provide flexibility for within-class variation.

## 5.3 Discriminatively Trained Part Based Models

### 5.3.1 Overview

We use an object detection system ([29]) that represents objects using deformable part-based models because of its ability to detect objects while permitting deformations which is useful in our problem where some parts of the tools may be occluded by hands or materials like paper and plastic. It builds on the pictorial structures framework ([30]). Pictorial structures represent objects by a collection of parts arranged in a deformable configuration. The individual part captures local appearance properties of an object while the deformable configuration characterize relative location of parts with respect to each other. A star-structured part-based model is defined by a root filter plus a set of parts filters and associated deformation models. The score for a model at a particular position ad scale within an image is the score of the root filter at the given location plus the sum over parts of the maximum, over placements of that part, of the part filter score on its location minus a deformation cost to penalize the deviation of the part from its ideal location relative to the root. The filter scores may be defined as dot product between the filter and a sub-window of the image. A single deformable model may capture some variations in the appearance of an object but they may not be rich enough for some categories. For example the category of bicycles which can be viewed in different poses (front versus side views). Therefore, the system described by [29] uses mixture models to deal with these variations. The score of a mixture model at a particular location and scale is the maximum over components, of the score of that component model at that location. A strong motivation for using this framework is its ability to detect objects using a mixture of models which permit intra-class variation. Fig. 5.1 shows one such component of the mixture model learned for markers.

In the following sections, we describe the models and how training and matching using such models is performed and also provide a description of the



(a) (b) (c)

Figure 5.1: Visualization of one of the component of marker model. The model is defined by (a) A coarse root filter (b) Higher resolution part filters (c) Spatial model for each part's location relative to the root

features used.

### 5.3.2 Models

The part-based model uses linear filters on feature maps which are arrays of feature vectors computed over a dense grid of locations in an image. Each feature vector describes a local image patch. To help define scores at different positions and scales in an image, a feature pyramid is used. The feature pyramid specifies feature maps at multiple scales in a fixed range and can be computed through repeating smoothing and sub-sampling. As one moves further down in the feature pyramid, the resolution at which a feature map is computed increases.
As mentioned in the previous section, a single model is composed of a coarse root filter which covers the entire object while part filters which are obtained at higher resolution cover smaller parts of the object. The part filters capture finer resolution features that are localized to greater accuracy when compared to features captured by the course root filter. This can be done by running the root filter at a higher location in the feature pyramid and placing the part filters at lower levels of the feature pyramid.
A model for an object with $n$ parts is formally defined as a $(n+2)$ tuple $(F_0, P_1, ..., P_n, b)$ where $F_0$ is the root filter, $P_i$ is the model for the $i$-th part and $b$ is a bias term. Each part model $P_i$ is defined by a 3-tuple $(F_i, v_i, d_i)$ where $F_i$ is the filter corresponding to that part, $v_i$ specifies an anchor position for the part relative to the root position and $d_i$ defines the deformation cost for the placement of the part relative to the anchor position $v_i$. The bias term $b$ is added to make the score of multiple models comparable when they are combined into a mixture model.
An object hypothesis $z$ specifies the location of each filter in the feature pyramid and is defined by a $(n+1)$ tuple $(p_0, p_1, ..., p_n)$ where $p_i = (x_i, y_i, l_i)$ specifies the level and position of the $i$-th filter. The score of a hypothesis $z$ is given by the sum of scores of a filter at its location in the hypothesis minus the deformation cost caused by displacement from its anchor position plus the bias.

$$score(z) \quad = \sum_{i=0}^{n} F_i \cdot FP(H, p_i) - \sum_{i=1}^{n} d_i \cdot FD(dx_i, dy_i) + b$$

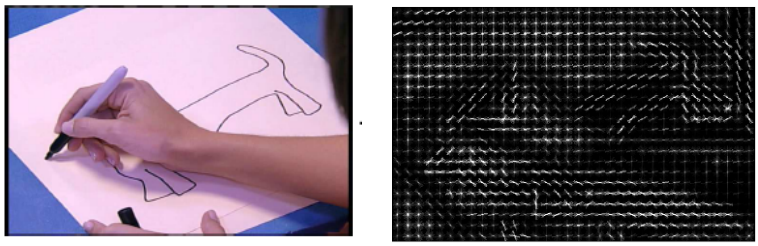
where $\mathrm{FP}(H, p_i)$ is a vector obtained as a sub-window from the feature pyramid $H$ of size equal to filter $F_i$ and top-left corner at point $p_i$, $(dx_i, dy_i)$ is the displacement of $i$-th part relative to the anchor and $\mathrm{FD}(dx_i, dy_i)$ is the deformation feature.
For detecting objects in an image, the overall score for each root location is computed as the best possible placement of the parts of the object. For calculating the best location for placing the parts, dynamic programming and generalized distance transforms are used. The method is efficient, taking $O(nk)$ time, where $n$ is the number of parts and $k$ is the number of locations in the feature pyramid. We refer the reader to [30] for more details. For object detection using a mixture model, the system finds root locations for each component independently using the method described above.

### 5.3.3 Training models

The training data consists of images and each image is annotated with bounding boxes for each object class present in the image. The annotations do not provide contain any information about the individual component labels and their part locations making it a weakly labeled setting. Since the part locations are not labeled, they are treated as latent variables during training.
For training using this weakly labeled data, [29] uses a latent variable formulation of MI-SVM ([5]) which they call *latent* SVM (LSVM). The latent SVM is trained using a coordinate descent approach. More details can be found in [29].

### 5.3.4 Features

For visual features to represent an object category, [29] uses the histogram of oriented gradients (HOG) features introduced in [23]. HOG features are count of occurrences of gradient orientation in localized portions of the image.
A pixel-level feature map is defined as an oriented edge map with $p$ orientation bins. For each pixel, one of the bins is chosen by discretizing the gradient orientation for that pixel. and its magnitude defines the edge strength in the feature map. Fig. 5.2 shows the HOG map for a sample image.



(a) (b)

Figure 5.2: HOG Map Visualization (a) Sample Image (b) HOG Feature Map

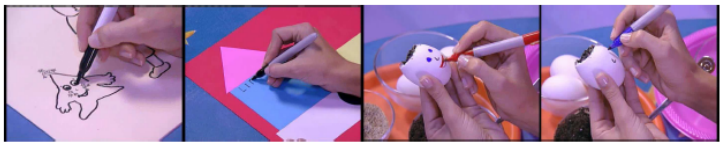
To achieve invariance, the gradient is normalized into 4 factors. In the experiments, the value for $p$ is 9 gradient orientations. This gives a HOG feature vector of 36 dimensions which has been defined using 4 normalizations of a 9 dimensional histogram over orientations. PCA is performed on these vectors and the dimensionality is reduced to 11 which leads to models with fewer parameters and faster learning algorithms.

## 5.4 Results

### 5.4.1 Object Class Selection

After an analysis of the data, the set of objects present in the PBS Sprout TV dataset was divided into two classes - "Tools" and "Stuff". The "Tools" class are objects which can be used to perform particular actions and they maintain a consistent visual appearance across the shots. The "Stuff" class objects may undergo transformation during an action and their visual appearance can undergo major changes during the course of the action. An example of the two classes would be *WritingTool* and *Paper* respectively as shown in Fig. 5.3. It can be observed that *Paper* undergoes transformations in color and shape while the general appearance of *WritingTool* remains the same.

(a)



(b)

Figure 5.3: Visualization of examples from two object classes (a) Paper (b) WritingTool

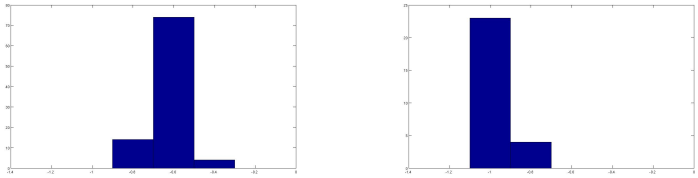
After these observations, the "Stuff" class was discarded since their appearance was subject to large variations in shape, color and texture. We concentrate on the "Tools" class which contains *WritingTool*, *Scissors*, *Brush* and *GlueBottle*.

### 5.4.2 Training

Images corresponding to the four object classes discussed in the previous section were obtained from external sources like ImageNet[1], LabelMe[2] and Google Images[3]. Training images included objects at different scales and orientations with occlusions present in some of them and may include multiple instances of the same object in one image. Each training image was labeled by a human annotator with bounding boxes for the object class present in each image. These images are then used to train individual models for the four classes using [29].

### 5.4.3 Experiments

Experiments were performed on "Full/Tight" shots from the PBS Sprout TV craft dataset. A single shot is a sequence of images. Running object detectors on each image of the shot to obtain object cues for the shot is an expensive task. Instead we run object detectors on a sample of the sequence of images in the shot. We run object detectors on every $k$-th image ($k = 5$ in this case) of the sequence. For each model, the top object hypothesis score for that image is computed. Performing this on the sample of images will lead to a set of detector scores for the shot. The presence of a particular object should lead to a set of higher scores for that object relative to the other object models. To obtain a summary of the object detector scores for a shot, we discretize the scores into 5 bins. The histograms define the "objectness" of the shot with respect to the particular object. Fig. 5.4 shows histograms obtained by running *WritingTool* detector on two shots, one of which contains a *WritingTool* while the other contains a *Scissor* object. It can be noticed that the histogram for the coloring shot has more peaks towards the right bin centers compared to the cutting shot.



(a) (b)

Figure 5.4: Histogram for *WritingTool* detector. The histogram bin centers represent detector scores while the percentage of images in a shot with scores in a particular bin are displayed on the y-axis (a) Shot where marker is being used for coloring paper (b) Shot where scissor is being used to cut paper

We compare the discriminative performance of the histograms for detecting particular objects in shots. Histograms were computed for "Full/Tight"

[1] http://www.image-net.org/
[2] http://labelme.csail.mit.edu/
[3] http://www.google.com/imghp

shots from 13 training episodes and used for analysis of detector performance on "Full/Tight" shots from 14 test episodes. Models were learned using multi-class logistic regression with $L_2$ regularization using publicly available implementation LIBLINEAR[4]. Fig. 5.5 shows the confusion matrix for running this classification on the aforementioned test episodes.

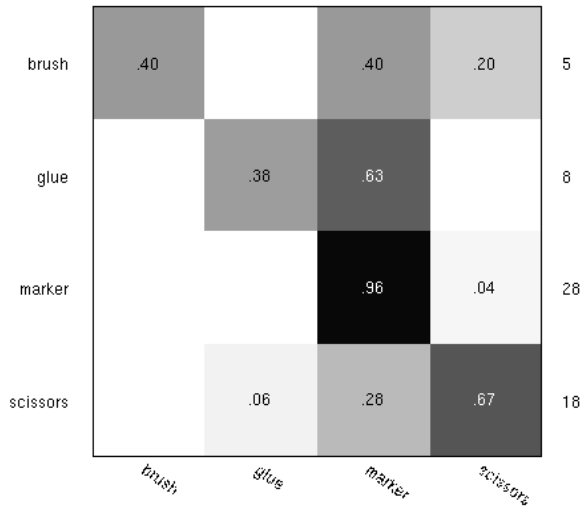



Figure 5.5: Confusion matrix for tool classification in shots. The number at the end of each row is the number of test shots for that object. It is observed that models for objects with higher frequency of occurrence perform better. *GlueBottle* is classified as $WritingTool$ in some cases because its conic shaped top is very similar to a $WritingTool$ like marker or pencil

The results of using detector score histograms for object classification in videos encourage us to use these histograms as object detector features in the system that will be used for joint modeling of textual and visual features (described in the next section).

[4]http://www.csie.ntu.edu.tw/ cjlin/liblinear/

# Chapter 6

# Attribute based descriptions

## 6.1 Motivation

In previous sections we have utilized language only to provide contextual information between objects and actions. We visually recognized the static entities in images (the objects) and the dynamic entities in video (the actions), and we mapped from visual space to language space, specifically to nouns and verbs. But clearly, these do not make up the full language space. Besides nouns and verbs, there are (a) prepositions that depict the temporal (e.g., before, after), spatial (under, on, in), and semantic relationship (e.g., with) among the nouns and verbs, (b) adverbs (e.g., fast, repetitive) and (c) adjectives (e.g., red, big, round) that depict the properties of the nouns and verbs. The prepositions, adjectives and adverbs are necessary ingredients to describe the scene and activity (i.e. to realize the map from vision space to language space). On the other hand, we need the prepositions, adjectives and adverbs to reason about the scene and activity and produce correct judgments from the noisy and ambiguous images and videos (from language space to vision space).

Within the workshop we took the first steps along these lines and worked on an approach to attribute-based object description. In principle, there are many advantages of using attributes for visual object description. Attributes are a higher level representation and could provide a way of going beyond what appearance based trained visual descriptors can achieve. Through language we can obtain organized knowledge about objects, that may not be possible to learn with current learning approaches. Even if learning can provide this knowledge, it may require large amounts of data. Using attributes we can transfer the knowledge of attributes and avoid supervised learning of object categories.

## 6.2 Overview of the approach

Previous work on attribute-based approaches (described in the next subsection) have dealt with images of single objects, and either did not consider the segmen-

tation of objects, or the problem was greatly simplified. In order fully utilize the language machinery information, we need to have a way of *segmenting* the scene, that is to locate the image regions containing objects in the scene. After we have found the regions we compute their attributes. Then we compute on these regions the properties of objects.

Segmentation is a very difficult task in general. Many state of the art computer vision algorithms allow us to compute optical flow, color models, edge maps, and segment regions. But within the context of manipulation, we can take an active, purposive approach, that makes segmentation feasible by introducing additional constraints. We are interested in the objects or tools held in the hands. Since tools are "extension" of the hands, they are tightly connected with the hands when they are in use. This makes the motion of a handheld tool very similar to that of the hand. Using this observation, we then developed the following approach. Using a CRF model we first detect the hands using color, edge and motion information, and the moving regions using motion and edges.. Then we remove the hand region from the moving regions to localize the tools. The implicit assumption follows from the observation that the tools which are relevant to the action must move with the hands.

The remainder of the section is organized as follows. Section 6.3 discusses related work. Section 6.4 presents our approach for segmenting the tools in the scene. Section 6.5 describes our implementation of a set of visual attributes and our ideas on visual object attributes and the design of the appropriate language tools in general. Finally Section 6.6 discusses further attribute based descriptions.

## 6.3 Related work

We review a few concepts from computer vision related to the visual processing in our approach.

**Attribute-based object description** has recently been addressed in a few studies. [31] learn to localize simple color and texture attributes from loose annotations provided by image search. [47] and [27] learn new object classes from known classes by transferring attribute descriptions. [43] use binary classifiers trained to recognize describable aspects of visual appearance in face verification. [84] propose learning models for obtaining color names from this noisy data. [89] extract color attributes from text description and use them to classify animal and plant species. [9] explore how to automatically discover attribute vocabularies and learn visual representations from unlabeled images and text data on the web.

**Image Segmentation** aims at partitioning an image into regions, where pixels in the same region are similar with respect to some representations, such as color, intensity, or texture. Graph based approaches (Normalized Cut [79]) and Partial Differential Equation based approaches (Level Set Methods [70]) are frequently used in computer vision. Recently, Conditional Random Field (see below) based approaches have been widely used in a number of difficult

multi-class image segmentation tasks with impressive results.

**Conditional Random Field (CRF)** is a discriminative model [46] that learns the conditional distribution of a class labeling given the input image data, and it has been widely used in a number of state of the art segmentation algorithms (see [80, 95, 91, 42]). CRFs extend the standard Markovian assumption in MRFs that nearby pixels are more likely to have the same labels by adding an additional conditional dependency on the input data. Such dependencies are modeled as unary and pairwise potential functions which are computed over a small pixel neighborhood clique. The final segmentation of the image is then obtained by determining the optimal labeling of the image pixels, which yields the lowest total energy over all the potentials. This can be efficiently accomplished using the well known $\alpha$-expansion algorithm ([13]).

## 6.4 Tool segmentation

Our approach to detecting the tool in the hand has the following steps:

1. Detecting and segmenting the moving hands in image frames using a trained CRF model. The hand segmentation result is denoted as $\mathbf{S}_h$.

2. Using the same CRF model applied on the computed optical flow, we obtain a segmentation of high optical flow regions which indicate moving objects in the scene that includes both hands and tools. This flow segmentation is denoted as $\mathbf{S}_f$.

3. The tool is then effectively localized by removing $\mathbf{S}_h$ from $\mathbf{S}_f$ followed by a few simple post-processing steps. The location of the tool is defined by a bounding ellipse and is denoted as $R_T$.

### 6.4.1 Conditional Random Fields (CRF)

The classical CRF model for segmentation amounts to setting up an energy function $E(x)$, with $x$ a label assignment over the image $I$ from the set of all possible labellings $L$, and it consists of unary and binary potentials as follows:

$$E(\mathbf{x}) = \sum_{i \in I} \phi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}(x_i, x_j) \tag{6.1}$$

The unary potentials $\phi_i$ encode the likelihood that a label is assigned to pixel $i$, and the pairwise potentials, $\phi_{ij}$, between adjacent points $i$ and $j$ encode consistency within segments.

Inference of the CRF involves minimizing Eq. 6.1 to obtain the labeling, $\mathbf{x}^*$:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbf{L}} E(\mathbf{x}) \tag{6.2}$$

and provides the segmentation of the image.

We encode two unary potential functions into Eq. 6.1

$$\phi_i(x_i) = \theta_{col}\phi_{col}(x_i) + \theta_{flow}\phi_{flow}(x_i) \tag{6.3}$$

where $\theta_{col}$ and $\theta_{flow}$ are the weighting parameters for the color, $\phi_{col}$, and optical flow, $\phi_{flow}$, respectively. $\phi_{col}$ is obtained from a Gaussian Mixture (GMM) color model, learned from training data over the CIELab color space. $\phi_{flow}$ is a unary prior that encodes the motion of the hand and the tool. $\phi_{flow}$ is obtained from a bimodal GMM of optical flow learned from the training data. By combining the color and flow potentials, we induce priors on hand-like regions that are moving.

By adjusting the weighting parameters $\theta_{col}$ and $\theta_{flow}$ we bias the segmentation either towards the color model (to get more hand-like regions) or the flow model (to get regions with higher flow). To obtain a segmentation that favors the hand color model, we set $\theta_{col} > \theta_{flow}$ such that the posterior likelihood computed from the hand color GMM is greatly increased. This means that pixels with color closer to the hand color will be labeled as hand, even if the flow in that region is small. Conversely, in order to obtain a segmentation that favors regions of high flow, we set $\theta_{flow} > \theta_{col}$.

In order to enforce consistent labels within segments, the pairwise potentials are defined as:

$$\phi_{ij}(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j \\ f(x_i, x_j) & \text{otherwise} \end{cases} \tag{6.4}$$

where $f(x_i, x_j)$ is an edge based contrast function defined over the image gradient, color and flow difference:

$$f(x_i, x_j) = \pi \exp(-\beta \left\| x_i - x_j \right\|^2) \tag{6.5}$$

where $\beta = (2 * \left\langle (x_i - x_j)^2 \right\rangle)^{-1}$ and $\langle \cdot \rangle$ represents taking the mean. $\pi$ is a constant parameter that is different for the gradient, color and flow features. Eq. 6.5 favors color and flow constancy within regions of similar color or flow by penalizing less the potentials when similar labels are assigned within a clique. This formulation had been widely used in several state of the art color segmentation algorithms ([12, 77]) with impressive results.

### 6.4.2 Segmenting Hands and Flow

The system must initially be trained to obtain the GMM color and flow models. This is done by randomly sampling 5 frames from the training sequence (each of length 100 frames) where binary labels of the color classes (hand and non-hand regions) and flow classes (large flow and small flow) are manually assigned (Fig. 6.1).

During the testing phase, a sequence of test image frames are presented to the algorithm which computes $\mathbf{S}_h$ and $\mathbf{S}_f$ in the following four steps as illustrated in Fig. 6.2:

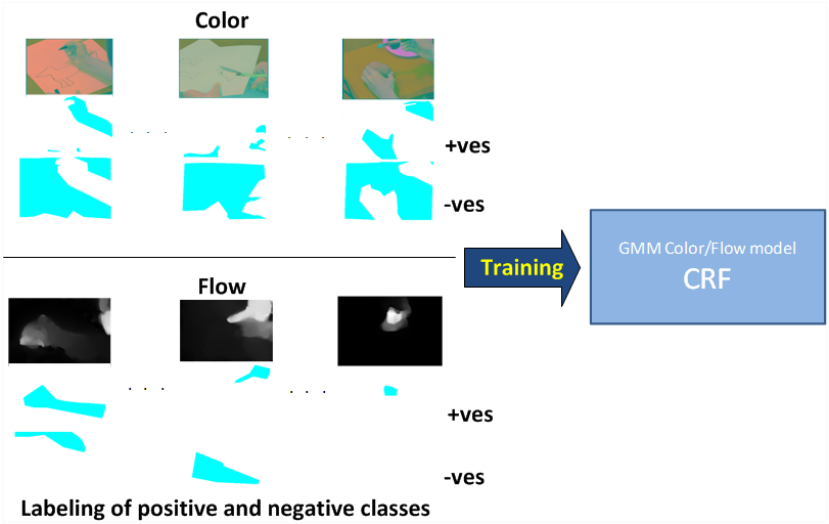



Figure 6.1: Training the GMM color and flow model. "+ves" and "-ves" denote the manually labeled positive and negative regions for the color (top) and flow (bottom) images.

1. Compute the optical flow $\mathbf{F}$ using the implementation of [14] between the previous and current image frames.

2. Compute the unary and pairwise potentials from Eqs. 6.3 and 6.4.

3. Obtain two energy functions $E(\mathbf{x})$ for the hands and flow by adjusting the unary potentials weights $\theta_{col}$ and $\theta_{flow}$ in Eq. 6.1. Since we still want consistent labels within the segments, the pairwise potentials $\phi_{ij}$ are the same in both cases.

4. Perform $\alpha$-expansion optimization on both energy functions using Eq. 6.2 to obtain $\mathbf{S}_h$ and $\mathbf{S}_f$.

### 6.4.3 Detecting Tools

The hand and flow segmentations, $\mathbf{S}_h$ and $\mathbf{S}_f$, provide the hand region and the regions of significant movement, respectively. Since $\mathbf{S}_f$ includes both the hand and the tools, the difference of the two segmentations will provide the tool location. In practice the following series of binary operations was used to obtain a good tool localization (Fig. 6.3):

1. Obtain the initial potential tool regions, $\mathbf{S}_r$, by removing $\mathbf{S}_h$ from $\mathbf{S}_f$ by: $\mathbf{S}_r = (\mathbf{S}_h \oplus \mathbf{S}_f) \cap \mathbf{S}_f$. This removes the hand regions while retaining only the regions in $\mathbf{S}_f$ that are connected to the hands. Note that $\mathbf{S}_r$ may be disjoint as the hand often occludes part of the tool.

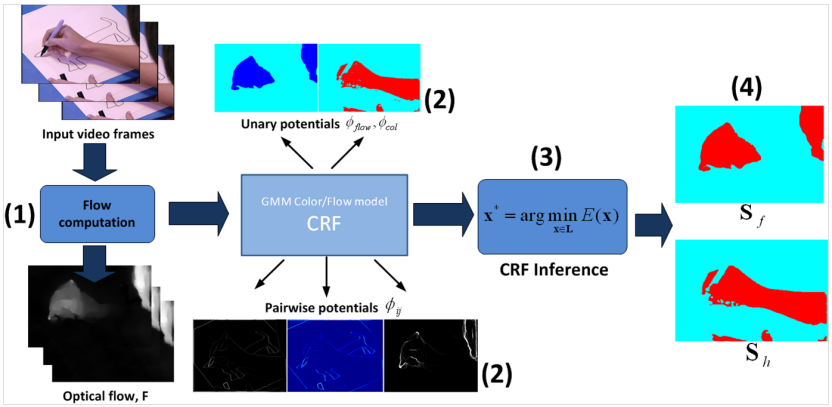



Figure 6.2: Segmenting hands and flow from a test image sequence. Labels with numbers correspond to the description in Sec. 6.2. (1) Compute the optical flow. (2) Compute the unary and pairwise potentials. (3) Perform inference to get the best labels to obtain the hand and flow segmentation in (4).

2. Compute an optical flow threshold, $t_f$, to accept the potential tool regions by taking the mean of the optical flow in the hand region: $t_f = \langle \mathbf{S}_h \cap \mathbf{F} \rangle$. Since the tool moves with the hand, its flow must be almost the same as the hand region

3. Process $\mathbf{S}_r$ to remove small spurious regions and accept only regions that have flow larger than $t_f$ and merge these regions together to form the filtered $\hat{\mathbf{S}}_r$.

4. Perform edge detection to obtain the edge fragments within $\hat{\mathbf{S}}_r$. By detecting only regions with significant edges, we remove false flow detections due to shadows (cast shadows from the moving objects create strong flow as well) since shadows have weak edges.

5. A minimum enclosing bounding ellipse is then fitted over the edge fragments which gives us the desired tool region, $R_T$ (visualized as the blue ellipse in Fig. 6.3).

### 6.4.4 Evaluation of tool localization

The algorithm for tool detection was evaluated over a subset of 10 video sequences (of 100 frames each) from the "Sprouts-TV" (STV) dataset that contain the five tool classes: {marker, paintbrush, crayon, scissors, glue}. All sequences had been hand-labeled with the tool's location to evaluate the accuracy in localization.

The video sequences were processed by the algorithm described above, which gave us the location of the tool, $R_T$, for every frame. We considered a detection

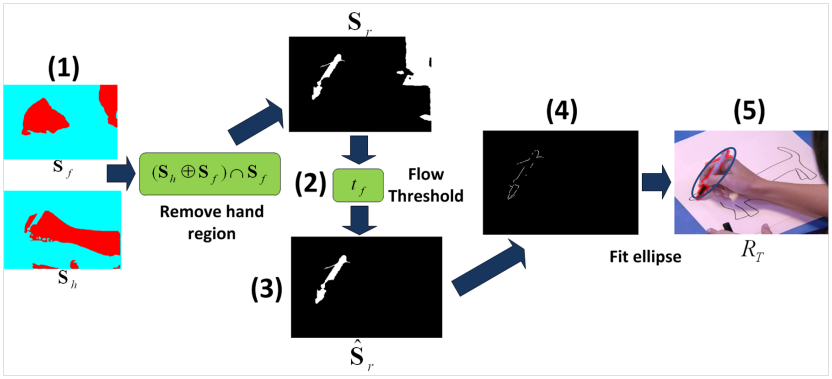



Figure 6.3: Localizing the tool region, $R_T$. The numbered labels correspond to the steps described in Sec. 6.4.3. (1) Input hand and flow segmentation from the test image. (2) Applying $t_f$ as a threshold to remove regions with flow that are different from the hand region to obtain (3). (4) Perform edge detection and (5) Fit the best ellipse over the edge fragments to locate the tool.

| Tool Class | $d_t(\%)$ |
|---|---|
| Marker | 89.1 |
| Paintbrush | 92.3 |
| Scissors | 87.9 |
| Glue | 84.0 |
| Crayon | 89.4 |

Table 6.1: Detection rates for the 5 tool classes in the STV dataset.

as 'correct' when a particular $R_T$ overlaps at least 50% and not more than 120% of the area in the ground-truth label. The upper bound is necessary so we do not falsely count detections that cover the entire image as 'correct'. $d_t$ was therefore computed as:

$$d_t = \frac{N_c}{N_S}, \tag{6.6}$$

where $N_c$ is the number of correct counts and $N_S$ is the total number of frames in the sequence (100 in all the sequences considered). The detection rates are given in in Table 6.1.

The average detection rate over all 5 classes is 88.5% which is reasonable given the the large variability of the tool classes in the dataset. This result is even more significant, since the proposed approach makes *no* implicit assumption on the tools that are detected – we only model the hand and motion, which are universal assumptions. This shows the feasibility of our approach in detecting tools in general situations where a hand action is occurring.

## 6.5 Computing attributes

As attributes of objects we can use color ('red', 'green', blue'), texture ('plaid', 'linear', pebbled'), surface reflectance ('metallic', 'matte'), shape ('round', 'elongated', 'square') and size. Computer Vision has the tools to compute descriptors which can capture these properties. We then need to develop classifiers based on these descriptors that are trained on databases of labeled images such as LabelMe. Color has previously been used, and is well understood [84]. The successful texture classifiers are based on modeling joint distributions of local neighborhoods [85], filter banks [51],[22] and fractal descriptors [92]. Surface reflectance is related to the material properties of an object [25], and can be classified with texture-like frequency operators. Shape attributes can be derived from the geometric properties of the segmented region. Finally size properties are available from comparing the object size with the size of the hand.

In future work we need to develop the language tools to relate objects with attributes. Color, and size are easily obtained from the web, such as Wikipedia and existing lexical databases (Wordnet, ConceptNet, etc.). The shape of complex objects may derived by finding the parts of an object and descriptions of the shape of parts and their geometric configuration. The texture of objects is a property of materials. We suggest then that texture attributes may be obtained in a hierarchical way by finding the constituent parts of an object and the materials these parts consist of. The texture properties of a small set of elementary parts will be defined manually. For example, we may find from language that a brush has bristles and bristles are made from hair, and hair has a linear structure. Figure 6.4 illustrates the idea.


Figure 6.4: Ongoing NLP work: Language tools to extract the attributes of objects from the web and lexical databases.

### 6.5.1 Experiments

To demonstrate the idea of attribute-based object description, we selected four attributes and evaluated them on the ten video sequences used in 6.4.4. These attributes are color, texture and two attributes of shape. Specifically, we dis-

tinguished between three color classes (white, silver and others), two texture classes (linear texture vs. others), the elongatedness of the region containing the object and a measure of convexity of this regions. Color was computed from color histograms in CIELab space, texture was computed using the Gist classifier [69], the elongatedness was derived from the ellipse fitted to the objects as the ratio of the major axis over minor axis, and the convexity was derived from the ratio of the area of the segmented region over the area of its convex hull. SVM classifiers were trained on labeled objects of internet images using the attribute category assignment as shown in Fig. 6.4. The average recognition rates for the color, texture and the two shape classifiers was found 87.0%, 93.3%, 91.0% and 93.3%, respectively, demonstrating the potential for attribute-based object recognition.

## 6.6 Discussion

The next step in combining language with vision for activity recognition in video is to utilize attributes of objects and actions. Attributes provide a higher level description and thus allow us to employ abstract visual descriptions. Objects are described by their adjectives and constituent parts. Actions can be can be described by properties that specify their style, direction, magnitude, speed, periodicity, by the position and the change of position of the hands and other body parts relative to the body and to objects, as well as by the transformation of objects during the action, for example the change in color (e.g., in paint) or shape (i.e., in cut).

Going from language to vision, we need to develop the linguistic tools such as the ones described before, to automatically obtain attributes of objects and actions so that we can search for the objects and actions through them. Going from vision to language we need to develop operators that can facilitate the extraction of the "red elongated" object or the "periodic" movement or "the object next to an object the system already knows". These operators are not just classifiers, but they involve necessarily also the segmentation. Visual segmentation has been considered a very difficult problem, and it does not appear that it can be solved in a purely bottom-up fashion. It appears that we need to introduce additional constraints through the process of attention, which is a higher cognitive process. Using attribute descriptions gives us a systematic way for addressing attention and introduce additional constraints into the visual segmentation. For example, if we know from language that we are looking for an object of certain color or texture, the vision system will pay attention to this color and texture, and the appropriate color or texture modeled can be introduced as priors into the optimization that delivers the segmentation. Coming back to our segmentation in Section 6.4.3, we used priors on the motion and color model of the hand, and implicitly we assumed we have knowledge about them. When we have all the tools in place, we can obtain information about these priors automatically from language.

# Part III

# Joint Modeling of Text and Visual Features

# Chapter 7

# Single Shot Action and Tool Recognition

## 7.1 Introduction

In the previous sections we have described several sources of information which all capture semantically meaningful aspects of action recognition: *Spatial temporal interest points (STIP)* capture local motion patterns, and are a popular and competitive representation for action recognition. *Hand pose* descriptors capture the orientation of the hand, which is highly correlated with what action is being performed, and more semantically meaningful than the bag-of-interest-points representation of STIP codewords. Finally, *object detectors* directly determine which tools are present, which also implicitly aids in the recognition of tool-correlated actions.

In this section we describe how to combine these sources of information for action *and* tool classification. We treat each zoomed-in shot in our dataset as a single example, which we wish map to a single action category (e.g., *Cut*) and a single tool category (e.g., *Scissors*). We assume a fixed finite list of actions and tools, and model the rest of the possibilities with action category *Other* and tool category *Other/None*.

We first discuss a straight-forward combination via standard supervised multi-class machine learning methods. Next, we propose a joint model for inferring about actions and tools, which can explicitly model the co-occurrence relations between different actions and tools. Finally, we show that we can incorporate prior domain knowledge into our joint model, which allows our system to scale to different and larger domains with minimal human supervision.

## 7.2 Independent modeling

Let $f_{hand}(x)$, $f_{tool}(x)$, and $f_{stip}(x)$ be our three sources of features, described above, for an example image $x$. Let $A$ be a set of action labels we are interested in applying. In our setting, $A = \{\ Color,\ Cut,\ Draw,\ Glue,\ Paint\ Other\}$. A standard way to model a multi-label classification task is with a linear function of the features for each class $a \in A$:

$$g^a(x) = w^a \cdot f(x)$$

where $g^a(x)$ is a score for example $x$ having label $a$, and $f(x)$ is a vector of features for example $x$, which can be some or all of $[f_{hand}(x); f_{tool}(x); f_{stip}(x)]$[1]. Using a labeled dataset, we can learn a set of parameters $w^a$ for each class $a \in A$ using a one-class-versus-rest type loss function. Depending on the exact form of the loss function, we can use off-the-shelf machine learning optimization algorithms to learn $\{w^a\}_{a \in A}$ — multinomial logistic regression using log-loss; Support Vector Machine (SVM) using hinge-loss and an $L_2$ penalty on the weights. At test-time, the most likely action label $a^\star$ can be obtained by

$$a^\star = \arg\max_{a \in A} g^a(x).$$

Similarly for example $x$ we wish to determine which tool $t \in T$ is being used where $T$ is the set of labels { Brush, GlueBottle, WritingTool, Scissors, None}. We can learn linear parameters $w^t$ for each tool, and classify with

$$t^\star = \arg\max_{t \in T} g^t(x) = \arg\max_{t \in T} w^t \cdot f(x)$$

## 7.3 Joint action-tool modeling

Clearly, different actions are highly correlated with the use of different tools. We would like to leverage this information in order to improve accuracy of both action and tool recognition. For example, a strong signal for a particular action (e.g., *Cut*) may help with an ambiguous signal for what tool is present (e.g., *Scissors*), and vice versa. When modeling actions and tools independently as in the previous section, this type of information can be learned implicitly by including object detector features when learning action recognition models, and action recognition features when learning tool recognition models.

However, this type of implicit modeling has several issues. For one, there is no way to jointly choose the best action-tool combination for a particular example. Two, there is no straightforward way to encode or learn priors on action-tool pairs. Finally, expressing these implicit features as a subspace in a larger feature space for which we learn a linear discriminative function may not be optimal.

[1] We use the convention that vectors in $d$ dimensions are $d \times 1$ (vertical), and use notation $[x; y]$ to mean the concatenation of vectors $x$ and $y$.

In light of this, we propose to model the joint probability distribution over possible actions and tools for each example: $p(A = a,\ T = t \mid x)$. We decompose this distribution into factors for how likely each action and tool are independently, as well as a term which explicitly encodes the likelihood of each possible (action,tool) pair.

**Incorporating Domain Knowledge**

One of the important things this model allows for is the incorporation of explicit, prior domain knowledge about action and tool co-occurrences into the model. This is beneficial for many reasons. Assuming that domain knowledge can be obtained at little or no cost—as is the case when automatically extracting it from web text—this significantly reduces the amount of human work labeling data. This is of critical importance when scaling up to larger or more varied domains. Furthermore, when working with a small dataset, the provided annotations might provide only a sparse, unreliable belief in action-tool co-occurrences. Substituting the annotations with domain knowledge automatically estimated from a much larger corpus (e.g., the web) can provide much more robust co-occurrence beliefs.

## 7.4 Action-Tool Conditional Random Field

We model $p(A,\ T \mid x)$ as a log-linear conditional random field ([46]):

$$p(A = a,\ T = t \mid x) = \tag{7.1}$$

$$\frac{1}{Z(x)} \exp\Big(w_A \cdot f_A(a, x)\Big) \exp\Big(w_T \cdot f_T(t, x)\Big) \exp\Big(w_{A,T} \cdot f_{A,T}(a, t)\Big) \tag{7.2}$$

where $w_A/f_A$ and $w_T/f_T$ correspond to action (respectively tool) parameters/features, and $w_{A,T}/f_{A,T}$ correspond to action-tool co-occurrence parameters/features. The term $1/Z(x)$ is a normalization constant which ensures the distribution sums to 1 over all (action, tool) pairs. Next we describe each term in our model, as well as inference and learning procedures.

**Unary terms $w_A \cdot f_A$ and $w_T \cdot f_T$:** We set $f_A(a, x) = [g^a(x);\ e_a]$ and $f_T(t, x) = [g^t(x);\ e_t]$, using the notation $e_i$ to denote the unit vector with a 1 in the $i^{th}$ dimension and zeros elsewhere. Thus our CRF features are the outputs of the independent action and tool models described in Section 7.2, in conjunction with class identity features $e_a$ and $e_t$ which allow the model to learn a prior likelihood of each class occurring (e.g., that *Draw* occurs more frequently than *Paint*). Using only these features constrains the model to only learn parameters $w_A$ and $w_T$ to balance the independent beliefs of different action and tool classes and class priors against the belief in action-tool co-occurrences, described next.

**Pairwise action-tool co-occurrence term $w_{A,T} \cdot f_{A,T}$:**

It is intuitive to think of the term $\exp(w_{A,T} \cdot f_{A,T}(a,t))$ in the form of an action-tool compatibility matrix: for every action-tool pair, it contains a corresponding real-valued score reflecting how likely the pair is to go together (e.g., *Cut-Scissors* should be very likely, *Cut-Brush* very unlikely). Thus we need to specify how to learn the entries of the action-tool compatibility matrix. We explore two different approaches.
*(1) Direct estimation of action-tool compatibility.* In this case we directly learn every entry in the action-tool compatibility matrix from our ground truth action-tool co-occurrences. To do this we express $f^{direct}_{A,T}(a,t) = [e_{(a,t)}; 1]$. The last component is a bias term to balance the values with the other terms in the CRF. The vector $e_{(a,t)} \in \mathbb{R}^{|A||T|}$ has a 1 in the $(a,t)^{th}$ dimension and zeros elsewhere, simply indicating the identity of the (action,tool) pair.
*(2) Action-tool compatibility via outside domain knowledge.* As discussed in the beginning of this section, there are several advantages to incorporating outside domain knowledge into our model instead of using training data. We assume domain knowledge comes in the form of $K$ co-occurrence matrices $C^k$, where $C^k_{a,t}$ is the real-valued entry in the $k^{th}$ co-occurrence matrix for action $a$ and tool $t$. The matrices can encode action-tool compatibility or incompatibility; the sign of the learned weights plus bias term can account for differing semantics. We incorporate these outside sources of information as a weighted combination of these co-occurrence matrices, where the learned weights reflect the usefulness / willingness to "trust" this knowledge compared to the other terms in the model. To accomplish this we set $f^{domain}_{A,T}(a,t) = [C^1_{a,t};\ \ldots;\ C^K_{a,t}; 1]$.

Note that because of the nature of our discriminatively-trained models, the action-tool co-occurrence values cannot be interpreted as joint probabilities, nor is the *direct estimation* approach as simple as computing ground truth co-occurrence frequencies.

**Inference:** Given the small number of variables (2) and state spaces ($\leq 10$) for each variable, inference can be performed quickly by brute force, enumerating and computing scores for all possible (action,tool) pairs. During testing we classify using the maximum a posteriori (MAP) decision

$$a^\star, t^\star = \arg \max_{a \in A, t \in T} p(A = a, T = t \mid x).$$

If we were to introduce more variables and structure into our model (e.g., modeling hand pose or temporal consistency), we could apply standard undirected graphical model efficient inference techniques, such as belief propagation dynamic programming.

**Learning** We learn parameters by maximizing the log-likelihood of our training data, with an additional regularization term. Let $w = [w_A; w_T; w_{AT}]$ be the parameters we wish to estimate. Assume we have a training set of $m$ examples which come with action and tool labels $\{(x^{(i)}, a^{(i)}, t^{(i)})\}_{i=1}^m$. The learning optimization problem is then

$$\textbf{minimize}_w \ \frac{\lambda}{2}||w||_2^2 - \sum_{i=1}^{m} \log p\Big(a^{(i)},\ t^{(i)} \mid x^{(i)};\ w\Big)$$

We use first-order gradient descent to optimize this convex function.

## 7.5 Results

We first analyze our proposed independent modeling of actions and tools described in Section 7.2, and then discuss our joint action-tool model.

To evaluate our techniques, we use the PBS Sprout TV craft dataset. Each of the 27 episodes is split into shots which are each a consecutive sequence of frames coming from one camera track. Furthermore, these shots can be divided into "Full/Tight" or not. We discard all non-Full/Tight shots as uninteresting, i.e., no action is present. The shot segmentation and classification can be done automatically with near perfect accuracy, as explained in for example [54], but for these experiments we use the perfect ground truth information to draw conclusions. We use ground truth annotation for learning and test evaluation: each shot is labeled with a single action and single tool class. We split the data into 13 training episodes and 14 test episodes, with the split chosen to have roughly the same number of examples of each action class in the training and test sets.

In this multi-class classification setting, we report *normalized accuracy*: the mean over all classes of the mean within-class accuracy. This performance measure is more robust to datasets where the number of examples of each class is very imbalanced, as in our data.

### 7.5.1 Independent modeling

Table 7.1 shows results for different multiple-action-classification settings, varying the number and types of classes, and features were used to learn the models. We learned models using multi-class logistic regression with $L_2$ regularization, using the publicly available implementation LIBLINEAR [2]. We found this to perform slightly better than linear, polynomial or Gaussian kernel SVM.

In the first column we examine a nearly balanced binary classification task between two intuitively distant actions, in terms of tools used, hand pose and motion pattern: *Cut* and *Draw*. We see that features $f_{tool}$ and $f_{STIP}$ alone do very well separating the data as expected. The hand features perform worse but better than random guessing.

Next we consider five-way classification between all action classes we have enough training data to model (i.e., more than five training examples). Again the hand features alone are the weakest cue, followed by tool features and STIP features. The combination of all three feature sources does better than any in isolation.

[2] http://www.csie.ntu.edu.tw/ cjlin/liblinear

| class (# in class) → <br> normalized acc.(%) ↘ | cut (18) <br> draw (20) | color (6) <br> cut (18) <br> draw (20) <br> glue (8) <br> paint (5) | color (6) <br> cut (18) <br> draw (20) <br> glue (8) <br> paint (5) <br> other (50) | brush (5) <br> gluebottle (20) <br> writingtool (28) <br> scissors (18) <br> none/other (48) |
|---|---|---|---|---|
| $f_{hand}$ | 63.3 | 27.8 | 20.5 | 23.5 |
| $f_{tool}$ | 91.7 | 42.9 | 37.1 | 48.8 |
| $f_{STIP}$ | 97.5 | 61.1 | 42.1 | 32.3 |
| $f_{hand} + f_{tool} + f_{STIP}$ | 97.5 | 67.1 | 44.0 | 46.0 |
| guess most frequent class | 50.0 | 20.0 | 16.7 | 20.0 |

Table 7.1: Independent modeling of actions and tools using logistic regression.

To obtain real end-to-end system results, we must also make a classification decision on the heavy tail of *Other* actions which occur infrequently and are extremely varied. Examples include "Crease", "Crackle", "Decorate","Shape","Sprinkle" etc.. In the third column we include this class, and see that performance suffers. This indicates the need to model more classes, or use other sources of information, like natural language, to narrow down the set of possibilities.

The general trend in these results is that the hand features perform the worst, but better than random; the tool features perform better, and the STIP features in isolation perform best of all. However, combining all three source of information can improve over STIP alone.

In the last column we perform a similar experiment on all shots, modeling the tool type rather than the action. Using the tool features works the best, while the hand features provide very little helpful information. The fact that STIP features work moderately well for tool classification and tool features work well for actions is empirical evidence that action cues can help determine tools and vice versa.

### 7.5.2 Joint modeling

Table 7.2 shows results from our CRF experiments. We found that explicitly modeling the co-occurrence of actions and tools either directly using our ground truth (column 2) or using domain knowledge (column 3) significantly helped results. For domain knowledge, we learned a weighted combination of the action-tool co-occurrence matrices obtained from the web, described in Chapter 3.

The action accuracy (row 1) remained nearly constant throughout experiments, but the tool accuracy (row 2) and accuracy in getting the correct tool *and* action together in the same example (row 3) increased significantly when modeling action-tool co-occurrence. Most importantly, these results demonstrate that it is possible to "plug in" domain knowledge as a substitute for ground truth information and get comparable performance gains over using no joint modeling.

| normalized accuracy (%) | no joint modeling | ground truth action-tool modeling | domain knowledge modeling |
|---|---|---|---|
| correct action | 50.9 | 50.8 | 50.8 |
| correct tool | 44.9 | 46.7 | 48.3 |
| correct action *and* tool | 28.0 | 40.7 | 37.8 |

Table 7.2: Joint modeling of actions and tools using a CRF.

## 7.6 Conclusion and Future Work

In this section we have described several models of increasing structure for action recognition. First, simple linear classifiers to model actions and tools using features based on hands, tool detectors, and motion interest points. Here we found that we can outperform the popular STIP motion descriptor by including the semantically meaningful hand and tool features. One possible extension of this work would be more sophisticated ways of combining the three sources of features. Currently, our models we simply concatenated the features to create one larger joint feature space. It may be beneficial to instead describe (kernelized) spaces for each feature source independently, and then learn to combine them as is done in multiple kernel learning.

Next, we propose a way to combine these models in a conditional random field which allows us to explicitly model the co-occurrence of action and tool classes. Our flexible model allows us to incorporate external domain knowledge as a replacement for ground truth co-occurrence information. This allows us to have more robust estimation when our ground truth information is very sparse. This is of critical importance when scaling up to many different domains in which actions and objects have interesting relationships.

# Chapter 8

# Temporal Modelling of Action Sequences

## 8.1 Introduction

In the previous chapters, we have discussed how to model the action within a single video clip. However, in real-world videos, such as those in the Sprout TV Handcraft Show, a whole video episode usually contains a sequence of actions. An example video episode is illustrated in Figure 8.1, which contains four actions: "draw", "color "cut", "thread". The texts associated to the videos, such as the transcripts of or online instruction provide us with cues for the temporal order of the actions in the video. The goal of this chapter is to investigate the methods to exploit the prior extracted from texts for temporal modeling of action sequence to improve the performance of action classification.

## 8.2 Method

From the example illustrated in Figure 8.1, we find that the order of actions within the transcript verb list provide us an important cue for the order of actions within the video episode. Formally, we hypothesize that the action bigram in the transcript implies the partial order of actions in videos, i.e., if there is an action bigram (v, w) in the text, the chance to find a corresponding action pair in the video sequence should be higher than otherwise. This hypothesis can be verified using the example video episode illustrated in Figure 8.1:

- The verb list extracted from the transcript
  - make make draw draw draw *draw color* cut tear use take
- The actions in the video
  - *draw color* cut thread

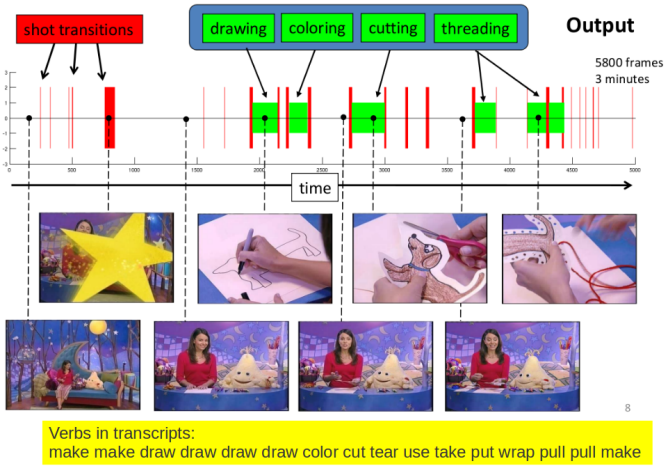



Figure 8.1: Example video "Babysitter's Animal Sewing Cards", PBS Sprout TV

Since the text and video are not strictly aligned in real videos, we further relax the action bigram to incorporate action pairs across up to two positions. The following example illustrates such a scenario:

- The verb list extracted from the transcript
  - use show *cut* tear *cut* make flatten take write
- The actions in the video
  - *cut cut* cut cut draw draw place place place

Similarly, such action bigrams can often be extracted from the online instructions. Thus, from the transcripts or the online instructions of training video episodes, we can compute the frequencies of action bigram. These frequencies provide us with the prior knowledge about the next action given the current action. Figure 8.2 and Figure 8.3 illustrate the frequencies of action bigrams computed from the transcripts and the online instructions of training video episodes respectively.

To integrate this knowledge with the probability of action classes obtained from individual video shots, we designed a chain CRF as illustrated in Figure 8.4. In this chain CRF, each hidden node (white circles) represents the action class label of a single video shot, and each observed node (grey circles) represents the observed visual features including motion and object features as discussed in previous chapters. The node potential is $p(A_i|X_i)$, the the probability of

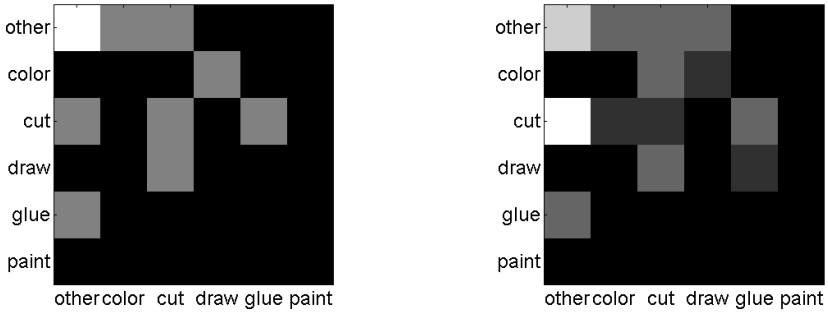



Figure 8.2: Frequencies of action bigrams in transcript in Sprout TV Handcraft Show

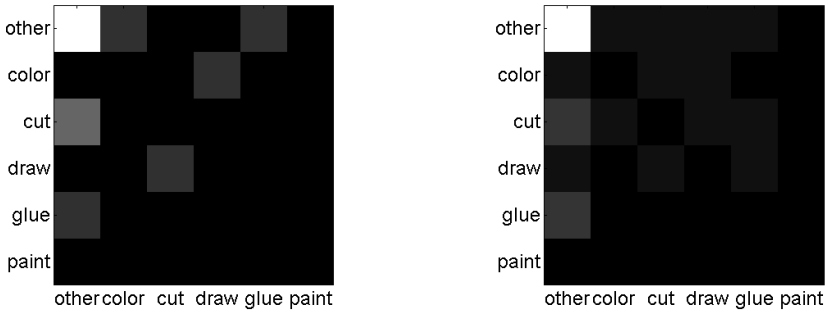



Figure 8.3: Frequencies of action bigrams in online instructions of Sprout TV Handcraft Show

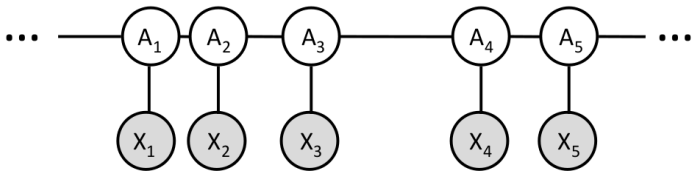



Figure 8.4: The chain CRF model for the action sequence in a video episode.

action class for video shot $i$ given the observed visual features $X_i$, and the edge potential is the exponential of the weighted frequency of the action bigram of consecutive video shot, $\phi(A_i, A_{i+1})$ . The conditional probabilities of the action classes in a video episode can be represented as follows:

$$p(A_1, A_2, ..., A_k|X_1, X_2, ..., X_k) = \frac{\prod_{i=1}^{k} p(A_i|X_i) \prod_{i=1}^{k-1} \alpha_i \exp\{\phi(A_i, A_{i+1})\}}{Z(X_1, X_2, ..., X_k))}, \tag{8.1}$$

where $Z(\cdot)$ is the partition function, $\alpha_i$ a parameter to weight the edge potentials. In the training stage, we need to estimate the weights $\alpha_i$ from the training corpus. Because the Sprout TV Handcraft Show dataset is very small and has only 13 training video episodes, we enforce $\alpha_i = \alpha$ to avoid over-fitting. The single parameter $\alpha$ can be estimated using cross-validation from training data. Given a test video episode, we first estimate $p(A_i|X_i)$ using algorithms described in the previous chapters. Then we substitute these values to Equation 8.1 and run the Viterbi Algorithm to find the most likely sequence of actions.

## 8.3 Experimental Results

In this experiment, we are interested in a relative performance change in action classification accuracy, i.e., with and without prior temporal knowledge obtained from transcripts and online instructions.In Table 8.1 we summarize the performance of average classification accuracy for the Sprout TV Handcraft Show dataset. These results show that the temporal knowledge extracted from transcripts and online instructions indeed improve the classification performance compared to the single shot model. Between two types of text sources, online instruction is slightly more helpful than transcripts. It is possibly because the transcripts are much noisier than online instructions as it contains large amount narration. Consequently, the verb list extracted from transcripts has more irrelevant verbs besides the action verbs we are interested in. The results also show that the relaxed bigram has no impact on the performance of action classification. By examining the details of trained chain CRF model, we find that the value of $\alpha$ is usually very small so the difference between the frequencies of action bigram and those of relaxed action bigram has little impact on the action classification performance.

| | |
|---|---|
| Single Shot Action Recognition using STIP (SVM) | 0.42 |
| previous + Tool + Hand Feature | 0.47 |
| Single Shot Joint CRF Model (STIP+Tool+co-occurrence of verb and tool) | 0.51 |
| Sequence Model CRF with temporal constraints extracted from transcripts (bigram) | 0.52 |
| Previous with relaxed bigram | 0.52 |
| Sequence Model CRF with temporal constraints extracted from online instructions (bigram) | 0.53 |
| Previous relaxed bigram | 0.53 |

Table 8.1: Overall action classification accuracy for the Sprout TV Handcraft Show dataset.

## 8.4 Summary and Future Work

In this chapter, we have described a method to integrate the temporal knowledge extracted from transcripts and online instructions with the single shot model and showed that the additional temporal knowledge is really helpful on action classification in the real video episodes that contain multiple actions.

The chain CRF model described in this chapter is a general model to incorporate any meaningful temporal constraints. Because of the limited size of Sprout TV Handcraft Show dataset, we only explore the relationship between adjacent nodes. Given a larger dataset, we can explore temporal relationship beyond adjacent nodes. We will further investigate this issue in the future.

# Bibliography


[1] Cmu quality of life grand challenge kitchen dataset. http://kitchen.cs.cmu.edu.

[2] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73:90–102, 1999.

[3] S. Ali, A. Basharat, and M. Shah. Chaotic invariants for human action recognition. In *Proc. International Conference on Computer Vision*, 2007.

[4] J. Allen. *Natural Language Understanding*. Addison Wesley, 2nd edition, 1994.

[5] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, pages 561–568. MIT Press, 2003.

[6] M. Baroni and S. Vegnaduzzo. Identifying subjective adjectives through web-based mutual information. In *Proceedings of KONVENS-04, 7th German Conference on Natural Language Processing*, pages 17–24, 2004.

[7] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[8] T. L. Berg, A. C. Berg, J. Edwards, M. Maire, R. White, Y. W. Teh, E. Learned-Miller, and D. A. Forsyth. Names and faces. *in submission*, 2010.

[9] T. L. Berg, A. C. Berg, and J. Shih. Attribute discovery and characterization from noisy web data. In *Proc. European Conference on Computer Vision*, 2010.

[10] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto. Recognition of human gaits. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 52–58, 2001.

[11] A. Bissacco, A. Chiuso, and S. Soatto. Classification and recognition of dynamical models: The role of phase, independent components, kernels and

optimal transport. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1958–1972, 2007.

[12] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary region segmentation of objects in n-d images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112 vol.1, 2001.

[13] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[14] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2010.

[15] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Comput. Linguist.*, 32(1):13–47, 2006.

[16] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[17] R. Chaudhry and R. Vidal. Recognition of visual dynamical processes: Theory, kernels and experimental evaluation. Technical Report 09-01, Department of Computer Science, Johns Hopkins University, 2009.

[18] R. Chaudry, A. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.

[19] R. L. Cilibrasi and P. M. Vitanyi. The google similarity distance. *IEEE Trans. Knowledge and Data Engineering*, 19(3):370–383, 2007.

[20] T. Cour, C. Jordan, E. Miltsakaki, and B. Taskar. Movie/script: Alignment and parsing of video and text transcription. In *ECCV*, 2008.

[21] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[22] O. G. Cula and K. J. Dana. 3d texture recognition using bidirectional feature histograms. *Int. J. Comput. Vision*, 59(1):33–60, 2004.

[23] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of IEEE CVPR*, pages 886–893, 2005.

[24] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, October 2005.

[25] R. Dror, A. Willsky, and E. Adelson. Statistical characterization of real-world illumination. *Journal of Vision*, 4:821–837, 2004.

[26] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Proc. International Conference on Computer Vision*, pages 726–733, 2003.

[27] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[28] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[29] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[30] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, January 2005.

[31] V. Ferrari and A. Zisserman. Learning visual attributes. In *Advances in Neural Information Processing Systems*, Dec. 2007.

[32] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, pages 363–370, 2005.

[33] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73:82–98, 1999.

[34] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE Trans. PAMI*, 29(12):2247–2253, 2007.

[35] I. Habernal and M. Konopík. Active tags for semantic analysis. In *TSD '08: Proceedings of the 11th international conference on Text, Speech and Dialogue*, pages 69–76, 2008.

[36] N. İkizler and D. A. Forsyth. Searching for complex human activities with no visual examples. *International Journal of Computer Vision*, 80(3):337–357, 2008.

[37] O. Jenkins, G. Gonzalez, and M. Loper. Interactive human pose and action recognition using dynamical motion primitives. In *International Journal of Humanoid Robotics*, volume 4, pages 365–385, 2007.

[38] L. Jie, B. Caputo, and V. Ferrari. Who's doing what: Joint modeling of names and verbs for simultaneous face and pose annotation. In *NIPS*, 2009.

[39] T. Kitani, Y. Eriguchi, and M. Hara. Pattern matching in the TEXTRACT information extraction system. In *Proceedings of the 15th conference on Computational linguistics*, pages 1064–1070, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[40] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

[41] D. Klein and C. D. Manning. *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, chapter Fast Exact Inference with a Factored Model for Natural Language Parsing, pages 3–10. MIT Press, 2003.

[42] P. Kohli, L. Ladicky, and P. H. S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[43] N. Kumar, A. Berg, P. Belhumeur, and S. Nayar. Attribute and simile classifiers for face verification. In *Proc. International Conference on Computer Vision*, 2009.

[44] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[45] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-01*, pages 282–289, 2001.

[46] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *In Proceedings of ICML*, pages 282–289, 2001.

[47] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[48] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[49] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[50] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[51] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Comput. Vision*, 43(1):29–44, 2001.

[52] J. Li and J. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *Transactions on PAMI*, 25(10), 2003.

[53] P. Liang, M. I. Jordan, and D. Klein. Learning semantic correspondences with less supervision. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, pages 91–99, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[54] R. Lienhart. Reliable transition detection in videos: A survey and practitioner's guide. *International Journal of Image and Graphics (IJIG)*, 1(3):469–486, 2001.

[55] D. Lin. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, Granada, Spain, May 1998.

[56] D. Lin and P. Pantel. DIRT - Discovery of Inference Rules from Text. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining*, pages 323–328, San Francicso, CA, 2001.

[57] H. Liu. Montylingua: An end-to-end natural language processor with common sense. Available at: web.media.mit.edu/ hugo/montylingua, 2004.

[58] H. Liu and P. Singh. Conceptnet - a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, Oct 2004.

[59] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[60] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[61] O. Maron and T. Lozano-Pérez. A framework for multiple instance learning. In *Adv. in Neural Information Processing Systems*, 1998.

[62] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[63] R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *Proc. International Conference on Computer Vision*, 2009.

[64] G. A. Miller. Wordnet - about us. http://wordnet.princeton.edu, 2009. WordNet. Princeton University.

[65] T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81:231–268, 2001.

[66] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104:90–126, 2006.

[67] R. Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):1–69, 2009.

[68] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. of IEEE CVPR*, pages 2161–2168, 2006.

[69] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, May 2001.

[70] S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 1 edition, October 2002.

[71] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.

[72] H. Poon and P. Domingos. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. ACL, 2009.

[73] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people and recognizing their activities. In *CVPR*, 2005.

[74] M. Raptis and S. Soatto. Tracklet descriptors for action modeling and video analaysis. In *Proc. European Conference on Computer Vision*, 2010.

[75] A. Ravichandran, R. Chaudhry, and R. Vidal. View-invariant dynamic texture recognition using a bag of dynamical systems. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[76] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of IJCAI-95*, pages 448–453, Montreal, Canada.

[77] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[78] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *Proc. International Conference on Pattern Recognition*, 2004.

[79] J. Shi and J. Malik. Normalized cuts and image segmentation. In *CVPR'97*, page 731, Washington, DC, USA, 1997. IEEE Computer Society.

[80] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[81] J. Sivic and A. Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(4):591–606, 2009.

[82] R. Socher and L. Fei-Fei. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2010.

[83] D. Tran and A. Sorokin. Human activity recognition with metric learning. In *Proc. European Conference on Computer Vision*, 2008.

[84] J. van de Weijer, C. Schmid, J. Verbeek, and D. Larlus. Learning color names for real-world applications. *Trans. Img. Proc.*, 18(7):1512–1523, 2009.

[85] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision: Special Issue on Texture Analysis and Synthesis*, 62(1–2):61–81, April 2005.

[86] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[87] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

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

[89] J. Wang, K. Markert, and M. Everingham. Learning models for object recognition from natural language descriptions. In *Proc. British Machine Vision Conference*, 2009.

[90] G. Willems, T. Tuytelaars, and L. J. V. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *European Conference on Computer Vision*, 2008.

[91] J. M. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *CVPR*, pages 37–44. IEEE Computer Society, 2006.

[92] Y. Xu, H. Ji, and C. Fermüller. Viewpoint invariant texture description using fractal analysis. *Int. J. Comput. Vision*, 83(1):85–100, 2009.

[93] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[94] A. Yilmaz and M. Shah. Actions sketch: A novel action representation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–989, 2005.

[95] P. Yin, A. Criminisi, J. Winn, and I. A. Essa. Bilayer segmentation of webcam videos using tree-based classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2010.

[96] Y. Zhai and M. Shah. Video scene segmentation using markov chain monte carlo. In *IEEE Transactions on Multimedia*, 8 (2006).

[97] J. Zhang, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 73, 2007.

[98] G. Zwieg and P. Nguyen. A segmental crf approach to large vocabulary continuous speech recognition. In *Proceedings of ASRU*, 2009.