Visual common-sense for scene understanding using perception, semantic parsing and reasoning

Somak Aditya[†], Yezhou Yang[*], Chitta Baral[†], Cornelia Fermuller[*] and Yiannis Aloimonos[*]
[†]School of Computing, Informatics and Decision Systems Engineering, Arizona State University
[*]Department of Computer Science, University of Maryland, College Park

Abstract

In this paper we explore the use of visual common-sense knowledge and other kinds of knowledge (such as domain knowledge, background knowledge, linguistic knowledge) for scene understanding. In particular, we combine visual processing with techniques from natural language understanding (especially semantic parsing), common-sense reasoning and knowledge representation and reasoning to improve visual perception to reason about finer aspects of activities.

1. Introduction and Motivation

Human visual cognition involves a combination of visual perception as well as reasoning with various kinds of associated knowledge (common-sense, background, domain, etc.). For example, consider the following scenario. A human puts tofu on a bowl. She then takes a knife and cuts the tofu in the bowl. Often, if the bowl is large, a nearby human observer will see tofu being put in the bowl and then see the "knife making a cutting action with respect to the bowl". However the observer, using her common-sense, is able to easily conclude that the second action is actually the knife cutting tofu. Now let us explore automating this cognition process and analyze what capabilities would be needed for an artificial system to emulate the human cognition in this example.

As we will elaborate in a later section a visual processing system developed by three of the co-authors (Yang, Fermüller, and Aloimonos 2013) can recognize and record the facts: *appears(put,tofu,bowl,1)* and *appears(cut,knife,bowl,2)*. From these facts the human is able to conclude *occurs(put,tofu,bowl,1)* and *occurs(cut,knife,tofu,2)*. For an artificial system to do that, it first needs to recognize that although *appears(cut,knife,bowl,2)* is perceived, one should not normally conclude *occurs(cut,knife,bowl,2)* from it. This can be done if the system has domain knowledge about knives and bowls being artifacts and the general common sense knowledge that one artifact does not cut another artifact. The system can also use the domain knowledge about the effect of action "put A in B" and conclude that at time point 2, the tofu is in the bowl. Then the system needs the visual common sense knowledge that if it appears that X cuts Y, but normally X should not cut Y, and there is something inside Y, then most likely X is cutting what is inside Y. Using this visual common sense knowledge and its earlier conclusion that (a) the tofu is in the bowl; (b) it appears that knife is cutting the bowl and (c) normally a knife should not cut a bowl, the system should then be able to conclude *occurs(cut,knife,bowl,2)*. In a later section we will show how our cognition system is able to do this.

The previous example is about how to use commonsense reasoning to correctly recognize a single action. In most scenarios, visual cognition of actions and activities involves cognition of complex actions or activities that consists of multiple short actions whose execution order satisfies certain constraints. Common-sense reasoning plays several distinct roles in this. *First*, while traditional visual processing records occurrences of some individual actions, commonsense reasoning can be used to reason with the occurrences of these actions *to make conclusions regarding finer aspects of the activity*. This is needed because often visual processing systems are based on classifiers that recognize specific actions or objects about which they are trained, and the number of such classifiers in a visual processing module may be limited. Using visual common-sense knowledge one can fill in some of the missing details and partly overcome the limited number of available classifiers. *Second*, *common-sense knowledge about the breakdown of some well known activities to individual actions* can be used in recognizing the higher level activities. *Third*, in the absence of the second kind of common-sense knowledge, the system may have to learn the decomposition of activities to individual actions from visual examples. However, it is difficult and time consuming to construct multiple visual examples that cover enough cases so as to learn a general decomposition of an activity. An alternate approach is to train using a few examples, and use common-sense knowledge to *generalize the learned decomposition* of an activity. Following are some examples elaborating on the above kinds of visual commonsense reasoning.

Consider an example of the occurrence of a complex action of *marking a line* on a plank of wood using a ruler and a pen. Our visual processing system is able to analyze a particular video of marking a line and record the occurrences of

the following basic actions.

```
occurs(grasp1,lefthand,plank,50,85).
occurs(grasp2,lefthand,ruler,95,280).
occurs(align,ruler,plank,100,168).
occurs(grasp3,righthand,pen,130,260).
occurs(draw,pen,plank,170,225).
```

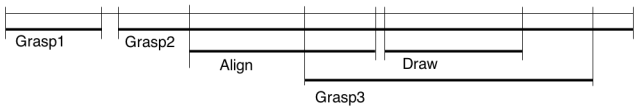


Figure 1: A timeline representation of the sub-activities of the activity *marking a line*

Intuitively, the meaning of the first action occurrence above is that "the lefthand grasped the plank during the time points 50 to 85." The meaning of the other action occurrences are similar. Now suppose the system is asked questions such as: (a) Which hand is being used in aligning the ruler; (b) Which hand is used in drawing; and (c) Is the ruler aligned when the pen is drawing on the plank? The visual processing systems may not have the modules to answer those questions directly; however, the action occurrences that are generated by the visual processing modules can be used together with common sense knowledge about actions and effects, about time and the domain of interest to answer the above questions. In a later section we show how ASP (Gelfond and Lifschitz 1988; Baral 2003) is used to express such common-sense knowledge and how its reasoning mechanism allows us to give the correct answers to the above questions.

Now consider this example from a different angle where the video is given as an example of how to recognize the marking of a line. Given just one example where the visual processing system obtains the above facts, one can come up with a rule that can recognize marking a line. Though, this rule would be too specific regarding which hands to use for the various actions. However, if we have the common-sense rule that under normal circumstances one can switch the hands all through an activity and achieve the same purpose, then we can come up with a more general rule to recognize marking a line. In a later section we show how such common-sense can be expressed using ASP and how we can use that to come up with a more general rule.

The rest of the paper is organized as follows. In the next section we present some additional background and discuss related work. In Section 3 we briefly describe our visual processing system. In Section 4 we discuss how visual common-sense reasoning can be used in improving visual perception. In Section 5 we discuss the role of common-sense reasoning in recognizing finer aspects of activities that may often be overlooked by the visual processing modules. In Section 6 we explore the role of common-sense reasoning in activity recognition. In Section 7 we conclude and discuss some future directions.

2. Background and Related Work

So far there has been very few works in the use of common sense in scene understanding. A few exceptions are (Wyatt, Philipose, and Choudhury 2005; Wang et al. 2007; Santofimia, Martinez-del Rincon, and Nebel 2012; Martinez del Rincon, Santofimia, and Nebel 2013). The focus in these works is on the use of common-sense reasoning in activity recognition. The first two works use simple common-sense models based on an average of four English words. The second two works mention various sources of common-sense knowledge and classify them as world knowledge, domain specific knowledge and expectations. We could not find any work that uses common-sense reasoning in the other aspects of scene understanding that we discuss in this paper, such as improving visual perception, recognizing finer aspects of activities and learning activity structures.

There is a large body of work in activity recognition. The early related work on this goes back to work in plan recognition (Charniak and Goldman 1993; Carberry 2001; Kautz 1987). In these works the focus was on recognizing what goals a sequence of action may be trying to achieve and plan recognition was treated as the opposite of planning. In recent years the problem of human activity recognition and understanding has attracted considerable interest in computer vision. Both visual recognition methods and non-visual methods using motion capture systems (Guerra-Filho, Fermüller, and Aloimonos 2005; Li et al. 2010) have been used. (Moeslund, Hilton, and Krüger 2006), (Turaga et al. 2008), and (Gavrila 1999) provide surveys of the former. There are many applications for this work in areas such as human computer interaction, biometrics, and video surveillance. Most visual recognition methods learn the visual signature of each action from spatio-temporal feature points (e.g, (Dollár et al. 2005; Laptev 2005; Wang and Suter 2007; Willems, Tuytelaars, and Van Gool 2008)). Work has focused on recognizing single human actions like walking, jumping, or running ((Ben-Arie et al. 2002; Yilmaz and Shah 2005)). Approaches to more complex actions have employed parametric models such as hidden Markov models (Kale et al. 2004) to learn the transitions between image frames (e.g, (Aksoy et al. 2011; Chaudhry et al. 2009; Hu et al. 2000; Saisan et al. 2001)). Among such works, the work by (Laxton, Lim, and Kriegman 2007) has a lot of similarities with our approach of activity recognition in this paper. However, as mentioned, in this paper we go beyond activity recognition to many other aspects of scene understanding.

3. Visual Processing System

This section gives a brief description of the visual processes, which have been designed specifically to be used with the AI reasoning modules. The input to our system are RGBD sequences. All sequences were recorded with one RGBD camera in a fixed location (we used a Kinect sensor). Human subjects were instructed to perform a set of manipulation actions with both objects and tools visible to the camera during the activity, and in all recordings only one human was present. To ensure some diversity, the actions were collected from two domains, a kitchen and a manufacturing environment. To further diversify the data set, we adopted two different viewpoints for each action set. For the kitchen actions, we used a front view setting; for the manufacturing actions, a side view setting.

The task of the vision system is to compute a sequence of the symbols in the form of (Subject, Action, Object, StartFrame, EndFrame), which are used as input to the grammar. In a nutshell, the symbols are computed as follows: A grasp type classification module provides a "Start of action" signal when the hand status changes from "Rest" to one of the three other types, and an "End of action" signal when it changes back to "Rest". During the such derived interval of the action the object monitoring and the segmentation-based object recognition module compute the "Object" symbol. The "Action" symbol is computed using as features the trajectory profiles provided by the the hand tracking module and the grasp type classification. These features produce "Action" symbols such as "Cut" and "Saw". The action "Assemble" did not have a distinctive trajectory profile, so we simply generated it when the "Cheese" merged with the "Bread" based on the object monitoring process. Since the perception system is not the main focus of the paper, only a very brief description of the different modules is given. For further details on the techniques, please refer to (Teo et al. 2012; Summers-Stay et al. 2013; Yang, Fermüller, and Aloimonos 2013; Yang et al. 2014).

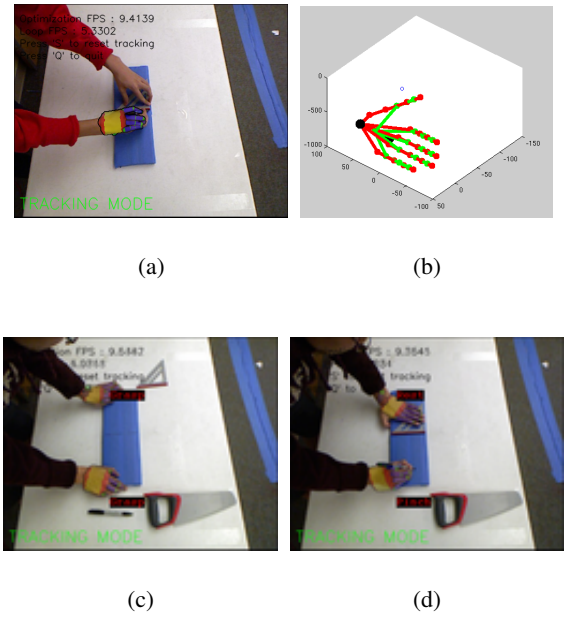
3.1 Hand Tracking and Grasp Type Recognition

We used the vision-based, markerless, articulated model-based human hand tracking system developed by (Oikonomidis, Kyriazis, and Argyros 2011) (http://cvrlcode.ics.forth.gr/handtracking/) to track the hands. This software uses a 26 degree of freedom model of the hand, but we only used a subset of the parameters. For the classification of the movements from the hand, we only used the center location of the hands. For the description of the grasp we reduced the dimensionality of the data, as described below, to classify the tracked hand-model into one of four different grasp types.

We collected training data from different actions, which we then processed as follows: A set of bio-inspired features (Tubiana, Thomine, and Mackin 1998), namely the arches of the fingers, were extracted. Intuitively, these arches are crucial to differentiate different grasp types. In each image frame, we computed the oblique and the longitudinal arches to obtain an eight parameter feature vector, as shown in Figure 2(a). We further reduced the dimensionality of the feature space using Principle Component Analysis and then applied k-means clustering to discover four types of grasp, which are: Rest, Firm Grasp, Delicate Grasp (Pinch) and Extension Grasp. To classify a given test sequence, the data was processed as described above and then the grasp type was computed using a naive Bayesian classifier. Figure 2(c) and (d) show examples of the classification result.

3.2 Object Monitoring and Recognition

Human actions involve objects. These objects are in the hands of the humans and during the manipulation often they may be occluded and only partly visible. Furthermore, the manipulated objects may change their geometry and even their topology during the action; they may be divided or two objects may merge. Thus, we need a process that monitors the objects being worked on. We use the method developed



(a) (b)

(c) (d)

Figure 2: (a) Example of the fully articulated hand model tracking, (b) 3-D illustration of the tracked model, and (c-d) examples of grasp type recognition for both hands (They are: 'Rest, 'Rest' (in c) and 'Rest' 'Pinch' in (d), shown with red letter in black boxes).

in (Yang, Fermüller, and Aloimonos 2013), which combines segmentation and tracking. This method combines stochastic tracking (Han et al. 2009) with a fixation-based active segmentation (Mishra, Fermüller, and Aloimonos 2009). The tracking module provides a number of tracked points. The locations of these points define an area of interest, and the center of this area provides the fixation point for the segmentation. The segmentation is addressed by minimizing for a closed contour surrounding the fixation point and segregating different colors, where the color model for the segment is derived from the region immediately surrounding the fixation point and it is compared to the regions surrounding the tracked points. The segmentation module segments the object and updates the appearance model for the tracker.

Figure 3 illustrates the method over time. The method is a dynamic closed-loop process, where active segmentation provides the target model for the next tracking step and stochastic tracking provides the attention field for the active segmentation.

For object recognition, our system simply uses color information. The color model is represented with a color histogram, which assigns one of m-bins to a given color at a given location. To be less sensitive to lighting conditions, the system uses the Hue-Saturation-Value color space with less sensitivity in the V channel ($8 \times 8 \times 4$ bins). Since the objects in our experiments have distinct color profiles, the color distribution model used was sufficient to recognize the segmented objects. For training, we manually labeled several examples from each object class and used a nearest k-

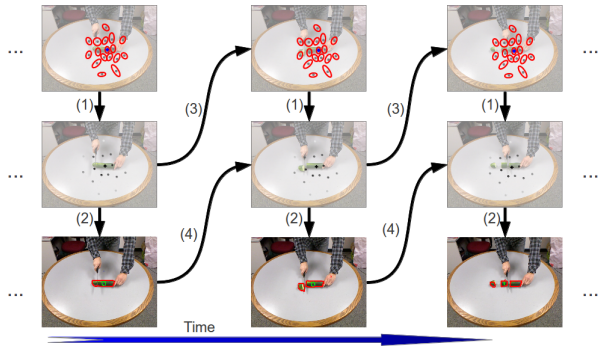


Figure 3: Flow chart of the active segmentation and tracking method for object monitoring: (1) Weighting; (2) weighted graph-cut segmentation; (3) point propagation and filtering; (4) updating of the target tracking model.

neighbours classifier.

3.3 Action Recognition

The grasp classification is used to segment the image sequence in time and also serves as a feature of the action description. In addition, our system uses the trajectory of the mass center of the hands to classify the actions. The hand-tracking software provides the hand trajectories (of the given action sequence between the onset of grasp and release of the object), from which our system computes global features of the trajectory, including the frequency and velocity components. The frequency is encoded by the first four real coefficients of the Fourier transform for each of the the x, y and z components of the trajectories. Together these components provide a 24 dimensional vector for the two hands. Velocity is encoded by averaging the difference in hand positions between two adjacent times tamps, which provides a six dimensional vector. These features (together with the grasp position) are then combined to yield the descriptor that the system uses for action recognition (Teo et al. 2012).

3.4 Towards unconstrained visual inputs

The perception system described above was developed to compute symbolic information (Subject, Action, Object, StartFrame, EndFrame) in a controlled lab environment, and we used Kinect data. Furthermore the recognition (of grasp type, objects, and actions) relies on the traditional hand-crafted features. However, it is possible also to generalize our approach to unconstrained video data. A recent work (Yang et al.) by three of the coauthors shows that with the recent developments of deep neural networks in computer vision it is possible to learn manipulation action plans in the format of (Subject, Action, Object, Temporal Order) from unconstrained demonstrations. However, due to the large variation and occlusions inevitable happening in unconstrained videos, our perception system computes wrong facts. In the next section, we will show an example how using answer set programming and a semantic parser can fix wrong visual detections.

4. Improving visual perception through background knowledge and common-sense reasoning

In this section we show how using answer set programming and a semantic parser built by two of the co-authors and their colleagues[1], we are able to make the correct conclusions regarding the occurrence of knife cutting tofu in the example described in Section 1.

Our Answer Set program below consists of facts of predicate "appears" obtained from the visual processing module. It has domain knowledge about knife and bowl being artifacts obtained from a semantic knowledge parser which takes phrases and parses them to knowledge graphs. We have the standard ASP rule to express effect of the action "put X in Y" and the inertia axiom for unaffected fluents. In addition we have three new kind of rules: (i) The first rule expresses that "normally an action that appears to have occurred indeed occurs"; (ii) the second rule defines an abnormality predicate that blocks the application of the previous rule; (iii) and the third rule defines when a particular appearance actually indicates that a slightly different action occurred.

```
appears(put,tofu,bowl,1).
appears(cut,knife,bowl,2).
artifact(knife).
artifact(bowl).
holds(in(X,Y),T+1):- occurs(put,X,Y,T).
holds(F,T+1):- holds(F,T), not nholds(F,T+1).
nholds(F,T+1):- holds(F,T), not holds(F,T+1).
occurs(A,S,O,T):- appears(A,S,O,T),
                          not ab(A,S,O,T).
ab(cut,S,O,T):- artifact(S), artifact(O).
occurs(cut,S,O,T):- appears(cut,S,O',T),
         ab(cut,S,O',T), holds(in(O,O'),T)).
```

We executed the above ASP program with addition of necessary domain predicates, such as time, and correctly obtained the answer set that contained *occurs(put,tofu,bowl,1), holds(in(tofu,bowl),2), occurs(cut,knife,tofu,2)*; meaning that the action "put tofu in bowl" occurred at time point 1, the action "cut knife with tofu" occurred at time point 2, and the tofu is in the bowl at time point 2.

5. Common-sense reasoning about finer aspects of activities

In Section 1, we listed the set of facts obtained by our visual processing modules analyzing a video about marking a line. We now present some commonsense rules written in ASP and show how those rules can be used in answer the following questions. (a) Which hand is being used in aligning the ruler? (b) Which hand is used in drawing? (c) Is the ruler aligned when the pen is drawing on the plank?

```
used(X,A1,T1,T2) :- occurs(A1,X,Y,T1,T2).
used(Y,A1,T1,T2) :- occurs(A1,X,Y,T1,T2).
used(H,A1,T1,T2) :- used(X,A1,T1,T2),
                    used(H,A2,T3,T4),
                    used(X,A2,T3,T4),
```

[1]This parser is available at http://www.kparser.org.

```
                grasper(H),
                T3 < T1, T2 < T4.
```

The first two rules above reason about which object is used in each of the actions and their time of use. Using the five occurs fact from the introduction section and only the first two rules above one can infer the following:

```
used(lefthand,grasp1,50,85).
used(plank,grasp1,50,85).
used(lefthand,grasp2,95,280).
used(ruler,grasp2,95,280).
used(ruler,align,100,168).
used(plank,align,100,168).
used(righthand,grasp3,130,260).
used(pen,grasp3,130,260).
used(pen,draw,170,225).
used(plank,draw,170,225).
```

The third rule is able to discover the use of graspers in actions that were not explicitly recorded by the vision processing system. Using the third rule and the facts that hands are graspers[2] we are able to infer the following two additional facts which answers the questions (a) and (b).

```
used(lefthand,align,100,168).
used(righthand,draw,170,225).
```

To answer question (c) we add the following rules to the ASP program which encode effect of actions, inertia axioms, and the question that is asked:

```
start(grasping,T1) :-occurs(grasp1,X,Y,T1,T2).
end(grasping,T2) :-occurs(grasp1,X,Y,T1,T2).
start(grasping,T1) :-occurs(grasp2,X,Y,T1,T2).
end(grasping,T2) :-occurs(grasp2,X,Y,T1,T2).
start(grasping,T1) :-occurs(grasp3,X,Y,T1,T2).
end(grasping,T2) :-occurs(grasp3,X,Y,T1,T2).
start(aligning,T1) :-occurs(align,X,Y,T1,T2).
end(aligning,T2) :-occurs(align,X,Y,T1,T2).
start(drawing,T1) :-occurs(draw,X,Y,T1,T2).
end(drawing,T2) :-occurs(draw,X,Y,T1,T2).
holds(aligned,T2+1) :-occurs(align,X,Y,T1,T2).
holds(drawn,T2+1) :-occurs(draw,X,Y,T1,T2).
holds(F,T+1) :-holds(F,T), not nholds(F,T+1).
nholds(F,T+1) :-nholds(F,T), not holds(F,T+1).
no :-    start(drawing,T1), end(drawing,T2),
      T1 < T, T < T2, not holds(aligned,T).
yes :- not no.
```

As expected, the program answers "yes" to question (c).

6. Common-sense reasoning in activity recognition

As we mentioned in Section 1 there are two main aspects to the use of common-sense reasoning in activity recognition. First, having common-sense knowledge about general structures of activities (i.e., how activities can be broken down to short actions and their timings), how to use that to recognize activities from the output of a visual processing system that gives us information about occurrences of short actions. Second, how to use common-sense in learning the general structures of activities from very few examples. Common to both these aspects is the need for a formal notion of the structure of an activity. We start with defining that.

Intuitively, an activity consists of smaller activities (which can be further broken down all the way upto short actions) and some constraints between their properties. Common type of constrains are temporal (the timing of the activities) and spatial (where the activities occur). While the notion of an activity has similarities to notions such as workflows, hierarchical task networks, complex actions (in languages such as Golog), and Petrinets, in this paper we start with a simple formulation that addresses the two above mentioned aspects of activity recognition. In particular, to automatically learn the structure of an activity, we need to have a simple enough notion for that. In addition we choose a simple notion of activity that makes it easier to make our main points of this section regarding the use of common-sense to recognize activities and to construct the structure of activities.

Thus, in this paper, an activity is defined in terms of a set of short actions, temporal ordering information about these short actions, and additional constraints about the actions and their parameters. Some of the predicates that we will use to describe the structure of an activity are: *component(SA,X,Y,A)*, *startsb4ov(SA1,SA2)*, *subinterval(SA1,SA2)*, and *before(SA1,SA2)*. Intuitively, *component(SA,X,Y,A)* means that the activity or short action *SA* with parameters *X* and *Y* is a component of the activity *A*. The intuitive meaning of *startsb4ov(SA1,SA2)* is that the start time of *SA1* is before the start time of *SA2*, they have some overlap, but *SA2* is not a subinterval of *SA1*. Similarly, the meaning of *subinterval(SA1,SA2)* is that the start time of *SA1* is after the start time of *SA2* and the end time of *SA1* is before the end time of *SA2*; and the meaning of *before(SA1,SA2)* is that end time of *SA1* is before the start time of *SA2*. Using this formulation the structure of the activity of marking a line can be expressed as follows:

```
component(g1,grasper1,plank,mark).
component(g2,grasper1,ruler,mark).
component(g3,grasper2,ruler,mark).
component(align,ruler,plank,mark).
component(draw,pen,plank,mark).
before(g1,g2).      subinterval(align,g2).
subinterval(g3,g2). subinterval(draw,g3).
startsb4ov(align,g3).
neq(grasper1,grasper2).
grasper1 in {righthand, lefthand}
grasper2 in {righthand, lefthand}
```

6.1 Activity recognition using ASP

Now to recognize that the *occurs* facts in the Introduction section corresponds to the activity "marking a line", we need to write rules that those occurs facts together satisfy the structure of the activity of marking a line. For that we need to define several rules that say when the occurs facts do not satisfy and then write the following rule:

```
satisfy :- not donotsatisfy.
```

Some example rules defining *donotsatisfy* is given below:

```
donotsatisfy :- component(X,Y,Z,A),
                not occuract(X).
occuract(X) :- occurs(X,Y,Z,U,V).
start(A,X) :- occurs(A,U,V,X,Y).
end(A,Y) :- occurs(A,U,V,X,Y).
```

[2]The knowledge that hands are graspers can be obtained from our semantic parser http://kparser.org when given sentences such as "The left hand was grasping the ruler.".

```
donotsatisfy :- before(A1,A2),end(A1,X),
                start(A2,Y), X >= Y.
```

The above is a very simple formulation. It can be straightforwardly generalized to compare against multiple activity descriptions by incorporating the activity names in the facts and rules. Uncertainty can be taken into account by allowing weights or probabilities to be associated with the occurs facts and using a matching probabilistic logic programming language (Baral, Gelfond, and Rushton 2009; Kimmig et al. 2011).

6.2 Learning Activity Structures

Our approach to learn the activity structures is based on the assumption that we may have only a few and sometimes only a single example from which to learn the structure. In that case one can not use standard inductive learning techniques that use a large number of examples and generalize based on that. Our approach is to first come up with a possible structure using the one (or a few) example(s) and then use common-sense knowledge about the domain to soften the constraints and generalize the structure. We illustrate this with respect to the "marking the line" example from the Introduction.

Knowing that the predicates that we want to learn are *component*, *statrsb4ov*, *subinterval* and *before*, we introduce "possible" version of these predicates which we call, *pcomponent*, *pstatrsb4ov*, *psubinterval* and *pbefore* respectively and write rules that define these predicates with respect to our example. Following are such rules.

```
pcomponent(A,X,Y,mark):-occurs(A,X,Y,T1,t2).
start(A,X) :- occurs(A,U,V,X,Y).
end(A,Y) :- occurs(A,U,V,X,Y).
pstartsb4ov(A1,A2):-start(A1,X), start(A2,Y),
                    end(A1,U), end(A2,V),
                    X  < Y, Y < U, U < V.
psubinterval(A1,A2):-start(A1,X), start(A2,Y),
                     end(A1,U), end(A2,V),
                     Y  < X,  U < V.
pbefore(A1,A2) :-  end(A1) < start(A2).
```

Using these rules together with the occurs facts, we obtain the following as part of the answer set:

```
pcomponent(g1,lefthand,plank,mark).
pcomponent(g2,lefthand,ruler,mark).
pcomponent(g3,righthand,ruler,mark).
pcomponent(align,ruler,plank,mark).
pcomponent(draw,pen,plank,mark).
pbefore(g1,g2).    pbefore(g1,align).
pbefore(g1,g3).    pbefore(g1,draw).
pbefore(align,draw). psubinterval(align,g2).
psubinterval(g3,g2). psubinterval(draw,g3).
pstartsb4ov(align,g3). psubinterval(draw,g2).
```

Next we use two kinds of common-sense knowledge to generalize the above and soften its constraints. First we use knowledge such as: (a) lefthand and righthand are graspers, (b) Ruler is an aligning artifact, and (c) pen is a writing instrument and the common-sense knowledge that normally in an activity one can replace one element of these categories by another as long as consistency is maintained. Using this knowledge we obtain the following:

```
component(g1,grasper1,plank,mark).
component(g2,grasper1,aligner1,mark).
component(g3,grasper2,aligner1,mark).
component(align,aligner1,plank,mark).
component(draw,winstr1,plank,mark).
neq(grasper1,grasper2).
grasper1 in {righthand, lefthand}.
grasper2 in {righthand, lefthand}.
aligner1 in {ruler}. winstr1 in {pen}.
```

Next we need to come up with a minimal set of facts about the predicates *before*, *subinterval*, and *startsbr4ov* that entail all the important *pbefore*, *psubinterval*, and *pstartsb4ov* facts. Domain knowledge can be used in determining which are important. In the absence of such knowledge we can consider all of them to be important, but even then we can still minimize the set of temporal constraints. This is achieved through the following steps: (i) we first enumerate all possible *before*, *subinterval*, and *startsbr4ov* facts; (ii) we then define *pbefore*, *psubinterval* and *startsbr4ov* in terms of *before*, *subinterval* and *startsbr4ov* and (iii) we then use the given important facts about *pbefore*, *psubinterval* and *startsbr4ov* as constraints and minimize the extent of *before*, *subinterval*, and *startsbr4ov* facts. Our ASP code implementing the above gives us *before(g1,g2), before(align,draw), subinterval(g3,g2), subinterval(align,g2), subinterval(draw,g3), and startsb4ov(align,g3)*, which is what we expected.

7. Conclusion, Discussion and Future Work

In this paper we gave a brief overview of how common sense knowledge and common sense reasoning can be used in various aspects of scene understanding starting with improving visual perception and leading to understanding of finer aspects of scenes, recognizing activities and learning activity structures. All through the paper we presented several snippets of ASP code.[3]

Our work uses more involved common-sense reasoning than used in existing papers (Wang et al. 2007; Martinez del Rincon, Santofimia, and Nebel 2013) and also goes beyond activity recognition, the focus in the existing papers. We also use more involved natural language analysis via our knowledge parser. While some of of our common-sense knowledge is hand written, some of it is obtained from our knowledge parser that integrates linguistic as well as world knowledge. In this paper we use ASP as our reasoning engine; however probabilistic extensions of ASP can also be used when reasoning with uncertainty is paramount. Our goal in the future would be to minimize hand written knowledge and develop ways to obtain knowledge from text and visual object repositories on demand. We have used a similar approach to address a class of Winograd challenge schemas. We also plan to develop an integrated end-to-end system that starts from visual processing, uses natural language processing to obtain necessary knowledge and then uses that knowledge in scene understanding. A feedback loop would further enhance this as one can then have a dialog mechanism that can iteratively improve scene understanding.

[3]Full version of the ASP codes is available at http://www.public.asu.edu/∼cbaral/papers/appendix-cs15.pdf

References

Aksoy, E.; Abramov, A.; Dörr, J.; Ning, K.; Dellen, B.; and Wörgötter, F. 2011. Learning the semantics of object-action relations by observation. *The International Journal of Robotics Research* 30(10):1229–1249.

Baral, C.; Gelfond, M.; and Rushton, N. 2009. Probabilistic reasoning with answer sets. *TPLP* 9(1):57–144.

Baral, C. 2003. *Knowledge representation reasoining and declarative problem solving*. Cambridge University Press.

Ben-Arie, J.; Wang, Z.; Pandit, P.; and Rajaram, S. 2002. Human activity recognition using multidimensional indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(8):1091–1104.

Carberry, S. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction* 11(1-2):31–48.

Charniak, E., and Goldman, R. P. 1993. A bayesian model of plan recognition. *Artificial Intelligence* 64(1):53–79.

Chaudhry, R.; Ravichandran, A.; Hager, G.; and Vidal, R. 2009. Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *Proceedings of the 2009 IEEE International Conference on Computer Vision and Pattern Recognition*, 1932–1939. Miami, FL: IEEE.

Dollár, P.; Rabaud, V.; Cottrell, G.; and Belongie, S. 2005. Behavior recognition via sparse spatio-temporal features. In *Proceedings of the Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 65–72. San Diego, CA: IEEE.

Gavrila, D. M. 1999. The visual analysis of human movement: A survey. *Computer vision and image understanding* 73(1):82–98.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R., and Bowen, K., eds., *Logic Programming: Proc. of the Fifth Int'l Conf. and Symp.*, 1070–1080. MIT Press.

Guerra-Filho, G.; Fermüller, C.; and Aloimonos, Y. 2005. Discovering a language for human activity. In *Proceedings of the AAAI 2005 Fall Symposium on Anticipatory Cognitive Embodied Systems*. Washington, DC: AAAI.

Han, B.; Zhu, Y.; Comaniciu, D.; and Davis, L. 2009. Visual tracking by continuous density propagation in sequential Bayesian filtering framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(5):919–930.

Hu, C.; Yu, Q.; Li, Y.; and Ma, S. 2000. Extraction of parametric human model for posture recognition using genetic algorithm. In *Proceedings of the 2000 IEEE International Conference on Automatic Face and Gesture Recognition*, 518–523. Grenoble, France: IEEE.

Kale, A.; Sundaresan, A.; Rajagopalan, A.; Cuntoor, N.; Roy-Chowdhury, A.; Kruger, V.; and Chellappa, R. 2004. Identification of humans using gait. *IEEE Transactions on Image Processing* 13(9):1163–1173.

Kautz, H. A. 1987. *A formal theory of plan recognition*. Ph.D. Dissertation, Bell Laboratories.

Kimmig, A.; Demoen, B.; De Raedt, L.; Costa, V. S.; and Rocha, R. 2011. On the implementation of the probabilistic logic programming language problog. *Theory and Practice of Logic Programming* 11(2-3):235–262.

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

Laxton, B.; Lim, J.; and Kriegman, D. 2007. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, 1–8. IEEE.

Li, Y.; Fermüller, C.; Aloimonos, Y.; and Ji, H. 2010. Learning shift-invariant sparse representation of actions. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2630–2637. San Francisco, CA: IEEE.

Martinez del Rincon, J.; Santofimia, M. J.; and Nebel, J.-C. 2013. Common-sense reasoning for human action recognition. *Pattern Recognition Letters* 34(15):1849–1860.

Mishra, A.; Fermüller, C.; and Aloimonos, Y. 2009. Active segmentation for robots. In *Proceedings of the 20009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3133–3139. St. Louis, MO: IEEE.

Moeslund, T.; Hilton, A.; and Krüger, V. 2006. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding* 104(2):90–126.

Oikonomidis, I.; Kyriazis, N.; and Argyros, A. 2011. Efficient model-based 3D tracking of hand articulations using Kinect. In *Proceedings of the 2011 British Machine Vision Conference*, 1–11. Dundee, UK: BMVA.

Saisan, P.; Doretto, G.; Wu, Y.; and Soatto, S. 2001. Dynamic texture recognition. In *Proceedings of the 2001 IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, 58–63. Kauai, HI: IEEE.

Santofimia, M. J.; Martinez-del Rincon, J.; and Nebel, J.-C. 2012. Common-sense knowledge for a computer vision system for human action recognition. In *Ambient Assisted Living and Home Care*. Springer. 159–166.

Summers-Stay, D.; Teo, C.; Yang, Y.; Fermüller, C.; and Aloimonos, Y. 2013. Using a minimal action grammar for activity understanding in the real world. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4104–4111. Vilamoura, Portugal: IEEE.

Teo, C.; Yang, Y.; Daume, H.; Fermüller, C.; and Aloimonos, Y. 2012. Towards a Watson that sees: Language-guided action recognition for robots. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation*, 374–381. Saint Paul, MN: IEEE.

Tubiana, R.; Thomine, J.-M.; and Mackin, E. 1998. *Examination of the Hand and the Wrist*. Boca Raton, FL: CRC Press.

Turaga, P.; Chellappa, R.; Subrahmanian, V.; and Udrea, O. 2008. Machine recognition of human activities: A survey.

IEEE Transactions on Circuits and Systems for Video Technology 18(11):1473–1488.

Wang, L., and Suter, D. 2007. Learning and matching of dynamic shape manifolds for human action recognition. *IEEE Transactions on Image Processing* 16(6):1646–1661.

Wang, S.; Pentney, W.; Popescu, A.-M.; Choudhury, T.; and Philipose, M. 2007. Common sense based joint training of human activity recognizers. In *IJCAI*, volume 7, 2237–2242.

Willems, G.; Tuytelaars, T.; and Van Gool, L. 2008. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proceedings of the 2008 IEEE European Conference on Computer Vision*, 650–663. Marseille, France: Springer.

Wyatt, D.; Philipose, M.; and Choudhury, T. 2005. Unsupervised activity recognition using automatically mined common sense. In *AAAI*, volume 5, 21–27.

Yang, Y.; Li, Y.; Fermüller, C.; and Aloimonos, Y. Robot learning manipulation action plans by "watching" unconstrained videos from the world wide web. To Appear.

Yang, Y.; Guha, A.; Fermuller, C.; and Aloimonos, Y. 2014. A cognitive system for understanding human manipulation actions. *Advances in Cognitive Sysytems* 3:67–86.

Yang, Y.; Fermüller, C.; and Aloimonos, Y. 2013. Detection of manipulation action consequences (MAC). In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2563–2570. Portland, OR: IEEE.

Yilmaz, A., and Shah, M. 2005. Actions sketch: A novel action representation. In *Proceedings of the 2005 IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, 984–989. San Diego, CA: IEEE.