ABSTRACT

Title of dissertation: PRODUCTIVE VISION: METHODS FOR AUTOMATED IMAGE COMPREHENSION

Douglas Summers Stay,
Doctor of Philosophy, 2013

Dissertation directed by: Professor Yiannis Aloimonos
Department of Computer Science

Image comprehension is the ability to summarize, translate, and answer basic questions about images. Using original techniques for scene object parsing, material labeling, and activity recognition, a system can gather information about the objects and actions in a scene. When this information is integrated into a deep knowledge base capable of inference, the system becomes capable of performing tasks that, when performed by students, are considered by educators to demonstrate comprehension.

The vision components of the system consist of the following: object scene parsing by means of visual filters, material scene parsing by superpixel segmentation and kernel descriptors, and activity recognition by action grammars. These techniques are characterized and compared with the state-of-the-art in their respective fields.

The output of the vision components is a list of assertions in a Cyc microtheory.

By reasoning on these assertions and the rest of the Cyc knowledge base, the system is able to perform a variety of tasks, including the following:

- Recognize essential parts of objects are likely present in the scene despite not having an explicit detector for them.
- Recognize the likely presence of objects due to the presence of their essential parts.
- Improve estimates of both object and material labels by incorporating knowledge about the typical pairings.
- Label ambiguous objects with a more general label that encompasses both possible labelings.
- Answer questions about the scene that require inference and give justifications for the answers in natural language.
- Create a visual representation of the scene in a new medium.
- Recognize scene similarity even when there is little *visual* similarity.

PRODUCTIVE VISION:
METHODS FOR AUTOMATED IMAGE COMPREHENSION

by

Douglas Summers-Stay

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:
Professor Yiannis Aloimonos, Chair/Advisor
Professor Cornelia Fermuller, Co-Advisor
Professor David W. Jacobs
Professor James A. Reggia
Dean's representative?

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Organization of This Work

This work explores the concrete steps needed to move towards this goal of a productive vision system. It includes novel algorithms to do scene parsing, activity parsing, and material parsing, compares these to existing techniques, and shows ways that they can be used to complement one another. It shows how the representation obtained from this can be used for description, comparison, and translation of scenes at a semantic level—the exact skills which, in a human student, would be said to show comprehension.

Chapter 2 is a look at how the field of teacher education has come up with tests for comprehension, especially comprehension of images. By grounding the problem in this way, I hope to forestall criticism that the computer vision and knowledge base system doesn't 'really' comprehend anything. This is probably a valid criticism, especially when it comes to qualia (my system's 'understanding' of 'the color red' does not include anything about what it is like to experience that color directly, nor is it clear whether giving a machine such an understanding is even possible.) The point of this section is to show that there is a well established practical definition of comprehension, and settled techniques for deciding whether such comprehension has occurred. In that sense, and perhaps that sense only, we can talk about image comprehension and expect to be able to make progress towards that goal.

The next three chapters cover the computer vision components necessary to

take input from a visual representation. They represent three categories of detection: detection of objects, detection of attributes, and detection of activities. Roughly speaking, these can be said to correspond to the concrete nouns, adjectives, and verbs respectively that would make up a scene description. The next chapters discusses the semantic network that ties these together, implemented within the Cyc knowledge base, and show how the system as a whole can be used for a variety of tasks demonstrating comprehension skills.

Supplement 1 is the book *Machinamenta: the thousand-year quest to build a creative machine*, published in December 2011. *Machinamenta* is only tangentially related to the rest of this work. It examines the history of the idea of a creative machine (considering both practical devices and fiction), and how our ideas about what 'creativity' and 'machines' are has changed over the centuries. Ultimately, the conclusion of the book is that without some ability to observe its own productions, comprehend their meaning and evaluate their impact, such machines will be fundamentally limited to being essentially kaleidoscopes. They would lack essential aspects of what we call creativity in other people. With that in mind, computer vision can be understood as important for creating computer artwork as image generation itself. Chapter 12 of *Machinamenta* is very different in character from the rest of the book. It contains some informed speculation about features a more advanced 'creative' machine might include. The 'image translation' in chapter 8 of this dissertation illustrates only one aspect of this: that human artists begin by understanding each of the objects they observe. It does not include the creation/observation feedback loop that is a critical part of creative work. For an example of what this would look like on the simplest level, see the Pareidoloop software by Phil McCarthy. [1]

Chapter 1: The Skills of Comprehension

1.1 Introduction

One of the key goals of computer vision research is known as "image understanding." This chapter lays out how teachers test for image "understanding" or "comprehension" by their students. While most of my effort in building the system has gone into creating new ways to parse objects, materials, and activities in scenes, a key component of the system is making use of a semantic knowledge base to tie observations to world knowledge.

The question of whether a computer program can "comprehend" anything at all is, perhaps, a philosophical one. Searle's famous Chinese Room argument is largely about understanding. Searle wrote "I have not tried to prove that 'a computer cannot think.' Since anything that can be simulated computationally can be described as a computer, and since our brains can at some level be simulated, it follows trivially that our brains are computers and they can certainly think." [2] So on Searle's view, thinking and understanding are possible for a computer, given the right architecture. Yet he feels that certain architectures are insufficient for comprehension, namely those without a semantic framework. "Having the symbols by themselves—just having the syntax—is not sufficient for having the semantics" and "lingusitic understanding requires at least a semantic framework." However, he gives no guidelines for recognizing when such a semantic network is present in a system.

Comprehension can mean many different things. Consider the famous hypothetical of Mary the color scientist [3]. Mary is colorblind but has learned everything there is to know about colors from textbooks. Can Mary be said to comprehend the meaning of the word *red*? If I comprehend "method acting," do I have the ability to do it? Since we have little idea how concepts are stored in the memory, it is at present impossible to know directly whether someone else has comprehended an instruction. To discover whether another person has comprehended something, we need to test whether they can perform tasks that would be difficult or impossible without comprehension. Fortunately, coming up with precise and accurate tests of whether comprehension has occurred has been an explicit goal of the educational establishment for years. This means that if we want to build a system for image comprehension, we have a well-defined target to aim at. If our system uses the same internal representation to pass several different already established tests of comprehension, then it can be said to have a practical understanding of the image.

In 1976, John McCarthy discussed the problem of a program being able to comprehend the following story from the New York Times. "A 61-year old furniture salesman, John J. Hug, was pushed down the shaft of a freight elevator yesterday in his downtown Brooklyn store by two robbers while a third attempted to crush him with the elevator car because they were dissatisfied with the $1,200 they had forced him to give them." He suggested that it should be able to answer questions like the following:

- Who was in the store when the events began?
- Who had the money at the end?
- Did Mr. Hug know he was going to be robbed?
- Does he know now that he was robbed?

Such a system would be useful in many ways, and would provide the kind of foundation needed for higher levels of visual reasoning: application, analysis,

synthesis and evaluation. Supplement 1, the book *Machinamenta* [9], discusses what is lacking in current efforts to build a creative machine. Comprehension, in this sense, is a necessary step before the synthesis and evaluation loop that is necessary for systems that can creatively extend their own domains.

There is also a kind of feedback effect between knowledge and comprehension. Without comprehension, it is impossible to organize knowledge. Every fact has to stand separately. As we try to build more and more extensive systems for vision, this will become a real problem. We would need to train a detector for every possible object and action. Using language, we are able to represent everything that exists or can be imagined using a limited set of symbols that can be recombined. The hope is that we can use a similar organization to perform the same trick with vision: to create a limited set of detectors that can, used in combination, describe the world as whole. I call this "productive vision," for two reasons. First, as in the phrase 'productive language,' the system is able to create novel structures by combination of predefined ones. The second meaning is a play off the mathematical term 'product': One can multiply, for example, the number of object detectors by the number of material descriptors to determine how many unique classes of objects the system is able to describe using just these components. Most current scene parsing systems, which treat both materials and objects as the same kind of label, would only be able to detect as many labels as the sum, rather than the product. Without some means of creating this combinatorial explosion in the number of potential scene descriptions a system can generate, any vision system will be fundamentally limited.

1.2 Teaching Reading Comprehension

The following is a sample reading comprehension test for the 3rd Grade Maryland School Assessment. [5]

"Being A Fish" by Russell E. Erickson

Would it be fun to be a fish? They are, after all, quite different from us.

Fish have no ears as we do. Their bodies are covered with thin, flat plates called scales. The only sounds they know are what they feel using certain scales along their sides. These are special scales called lateral lines.

We get oxygen from the air by using our lungs. Fish get oxygen from the water by using the gills on the sides of their heads. We can play in water and on land, but fish must stay in the water all the time.

Fish never get hot or cold. They are called cold-blooded because they are always the same temperature as the water around them. That means they have no need for hot soup, or cold lemonade, or cozy blankets, or cool sandals.

All in all, it's probably more fun being us.

Read the story "Being a Fish" and answer the following questions.

This story mainly tells _____.

1. how fish are different from people
2. how many kinds of fish there are
3. where fish can be found
4. how fish swim

You would most likely find an article like this in a book about _____.

1. farm animals
2. water fun
3. sea life
4. fishing

How do fish hear?

1. With ears
2. By feeling
3. By tasting
4. By smelling

These questions are designed to test for reading comprehension. They are written in such a way that simple syntactic strategies will fail. For example, a beginning English speaker might be able to answer the simpler question "Do fish have ears as we do?" by recognizing the sentence "Fish have no ears as we do" contains the same phrase with the addition of the word "no" and correctly answer "Fish have no ears as we do" without understanding the meaning of the words

"fish" or "ears." Searle supposed that more elaborate versions of such strategies (too elaborate for a human learner, perhaps, but not for a computer or a human with extensive role books to follow) might be able to give the false appearance of comprehension.

In both artificial intelligence and elementary school teaching, the problem of comprehension has been primarily associated with understanding text. Turing's "Imitation Game," for example, assumes a machine whose only input source is a stream of natural language text. The reasons for this are somewhat different between the two fields, however. Image understanding is largely not taught to students because it is too easy—by the time students are in school, most have acquired the ability to look at a scene, recognize the objects and activities in the scene, and make inferences regarding it without any formal training. In the field of AI, however, image comprehension has been considered too difficult. We are only in the past decade getting to the point with our image recognition and scene parsing programs that a few objects can be reliably recognized in arbitrary natural images.

1.3 Teaching Image Comprehension

The educational theory and practice involving image interpretation is mainly concentrated on those students who are poor readers. For those learning to read for the first time or learning a second language, the task of reading a paragraph and then reading questions about the paragraph requires decoding two separate texts (the passage to be read and the questions.) Interpreting images is a relatively easier

task for humans and is learned at an earlier stage of development. For example, [6], suggests that strategies for reading comprehension can be taught to students by first presenting them with the easier task of interpreting images. The answers to some questions are "right there" in the image. Others questions require bringing other knowledge about the world to answer them. A third set of questions about images requires inference combining information from various parts of the image.

In [7], children's ability to comprehend narrative of wordless picture books was tested. Questions were divided into the categories "explicit" and "implicit." Explicit questions included "Who are the people in this story?" "Where does this story happen?" "What happened here? Why did this happen?" Implicit questions, on the other hand, could not be answered with the information in the picture alone, but required additional information the student was expected to already have. These questions include "Tell me what the people are feeling in this picture" and "what do you think happens next." Similarly, in [8] assessment included questions such as "names objects and characters," "describes actions," "labels emotions" and more advanced inference such as "identifies motive," "predicts what will occur next" and "provides summary to story."

All these tests are predicated on the idea that "comprehension" can be tested by asking the subject to perform certain tasks. If the subject performs those tasks successfully, comprehension is said to have taken place. To this way of thinking, comprehension is a set of testable skills.

One of the most popular and long-lasting educational theories is known as "Bloom's Taxonomy." [10] The taxonomy was first published in 1956 in *A Tax-*

onomy of Educational Objectives—The Classification of Educational Goals, edited by Benjamin Bloom. The taxonomy was an attempt by college examiners and education professionals to categorize skills that could be taught in schools and to clarify more precisely what was meant by terms like "really understand," "internalize knowledge," "grasp the core or essence," or "comprehend." [11]. At the lowest level of this heirarchy was placed knowledge, because it was a necessary precursor to any other kind of cognition.

Revised Bloom's Taxonomy from [10]

- Knowledge: Retrieving, recognizing, and recalling relevant knowledge from longterm memory. Comprehension: Constructing meaning from oral, written, and graphic messages through interpreting, exemplifying, classifying, summarizing, inferring, comparing, and explaining.
- Application: Carrying out or using a procedure through executing, or implementing.
- Analysis: Breaking material into constituent parts, determining how the parts relate to one another and to an overall structure or purpose through differentiating, organizing, and attributing.
- Evaluation: Making judgments based on criteria and standards through checking and critiquing.
- Synthesis: Putting elements together to form a coherent or functional whole; reorganizing elements into a new pattern or structure through generating, planning, or producing.

Some skills that demonstrate knowledge are

- define
- identify
- know
- label
- list
- match
- name

This taxonomy was developed in part as a way of helping teachers to approach the problem of education on a deeper level than "rote" learning. For the purposes of the taxonomy, knowledge was defined as 'little more than the remembering of the idea or phenomenon in a form very close to that in which it was originally encountered." [11] This is illustrated by the story of John Dewey, who asked another teacher's class, "what would you find if you dug a deep hole in the earth?" When none of the children were able to answer after repeated questioning, the students' teacher explained to Dewey, "you're asking the wrong question." She asked the class, "What is the state of the center of the earth?" to which the class replied unanimously "Igneous fusion." [11] Most of what is being done in the field of computer vision today is at this level. Vision systems can successfully recognize objects in the scene because they (implicitly or explicitly) recall similar objects that they have seen before. At the next level up in Bloom's hierarchy is comprehension. To comprehend something takes more than knowledge. These skills include the ability to

- interpret

- extrapolate
- summarize
- compare
- describe
- translate

The handbook explains "Although the term 'comprehension' has been frequently associated with reading, e.g., reading comprehension, the use to which it is being put here is a somewhat broader one in that it is related to a greater variety of communications than that encompassed by written verbal materials. In another sense, the use of the term here is somewhat more limited than usual, since comprehension is not made synonymous with complete understanding or even with the fullest grasp of a message. Here we are using the term "comprehension" to include those objectives, behaviors, or responses which represent an understanding of the literal message contained in a communication." [11] In other words, 'comprehension' (as the term is defined for the purposes of the taxonomy) includes not just reading comprehension, but also comprehension of images; and it is not the fullest level of understanding that can be achieved.

1.4 Automated Image Comprehension

With this precise definition of 'comprehension' in hand, and with well-accepted examples of what kinds of tasks need to be performed in order to demonstrate that this level of understanding has been met, we can approach the problem of automated

comprehension in a way that hopefully won't dissolve into arguing semantics. In order to say that our system comprehends an image or video, it should be able to perform those kinds of tasks successfully.

There has already been quite a bit of work in solving these problems in the area of natural language understanding. The problems and solutions that have been found in attempts to comprehend text can inform similar efforts with images. For example, attempts at recognizing textual entailment eventually came to the conclusion that some form of world knowledge couldn't be avoided—even the simplest inferences required a body of information about things we consider perfectly obvious. The vision community has the advantage of coming late to the party. We don't need to develop an analogous knowledge base for understanding images but simply make use of the knowledge base already being developed for understanding text. Similarly, attempts to translate images from one modality to another using "image analogies" [12] use a shallow statistical correspondence between a photograph and an oil painting. Though the results look impressive, there is a ceiling above which such techniques can't go. Purely statistical language translation similarly is only capable of performing well in situations where large bodies of translated text already exist.

When a sentence is understood at the simplest level, one has recognized all the words of the sentence and the role they play. This process is called 'parsing' the sentence. (The terminology was only later adopted into computer languages.) In computer vision papers, the phrase "scene parsing" means labeling all of the pixels in an image. One can imagine a more comprehensive parsing, however, that includes

not just the nouns recognized in scene parsing, but also parsing out adjectives describing those nouns and verbs describing the action in the scene. A full parse of a video sequence would identify the subjects, actions, objects of the actions, and the adjectives and adverbs describing these.

The representation obtained by such a parsing should be useful for the comprehension skills listed above: A description or summary of the video might be a straightforward English sentence using these parsed terms. A comparison could find other videos which were similar not just in surface appearance (as image matching techniques do) but similar in what they are showing, similar in meaning. Translation could mean representing the image in a new medium— representing a photograph as a children's drawing, for example.

These are the tasks that are attempted in chapters 6 through 8. Chapters 3 through 5 discuss various tools developed to take images or video as input and return simple labels defining what the regions of the image contain. Speaking very loosely, they are tools to detect and mark out some of the concrete nouns, adjectives, and verbs in the scene. These will make it possible to make "sentences" in a predicate logic language that describe the scene in a way that makes possible the completion of these tasks.

Chapter 3: Visual Filters for Scene Understanding

3.1 Background of Object Recognition and Scene Parsing

Object recognition in the early years of computer vision (from about 1965-1990) mainly concentrated on matching a known arrangement of object contours, matching a shape from a particular perspective. Beginning in the 1990s, researchers [13] saw some success with *appearance-based models*, using empirical collections of examples of a class to define a manifold in some image space, and testing whether points (representing object appearance) lay on that manifold. It was immediately recognized that such techniques could be applied in a sliding window, sometimes at various scales, in order to locate the detected objects within the image.

Around 2002, material and object detection methods were first employed to create a labeled bitmap corresponding to the pixels of the input image. (See [14] and [15].) This went beyond the appearance-based object recognition techniques, which typically returned a rectangular box around the detected objects and elevated recognition of background classes to to the importance of recognition of the foreground. The term "Scene Parsing" was introduced by [16] in 2007. These techniques all required some form of graphical models which used belief propagation or other graphical models to decide where to put the boundary between image labels.

In 2009, a member of our research group, Justin Domke, wanted to demonstrate the effectiveness of the expectation maximization techniques[1] he was experimenting with. For purposes of comparison, he came up with a graphical combination of classifiers that didn't use these techniques at all, focusing on simplicity and performance. To his surprise, this method performed at nearly the level of the more sophisticated techniques. "Visual filters" are what we call the various extensions of this basic method, for reasons that will be explained below.

3.2 System Architecture

A very simple way to classify all the pixels in an image would be something like the following:

1. Collect statistics on the frequency that RGB values of each pixel of images in the training set correspond to particular labels.
2. Assign those label probabilities to each pixel of images in the test set independently, based on its color.

This approach would unfortunately be very inaccurate. It would be impossible to tell, for example, whether a brown pixel was dirt, fur, or a reddish surface in greenish light. A better approach might consider a larger patch of pixels surrounding the pixel we wish to label. (This describes a nonlinear convolutional filter that considers a window surrounding a pixel in order to assign it a value.) This would do a better job of labeling, because similar patches are more likely to come from similar contexts than individual pixels. As the support of the filter grows, the number of

[1] a way of solving such graphical models.

pixels involved quickly becomes unreasonable; a square RGB support width of just ten pixels creates a 300-dimensional feature. It would also be difficult to find a distance measure that worked well at finding similar raw image patches at large scales. Instead of patches, any kind of local feature could be used to try to match similar regions from the training set: color histograms or gradient histograms, a bank of oriented Gabor features, etc... In [17] for instance, this method is used directly as a scene parsing technique, using dense SIFT features from globally similar images.

One problem with this approach is that local windows, even with fairly wide support, don't provide enough context to differentiate between similar areas from differently labeled regions no matter what features are used. In order to solve this, we need to take advantage of the fact that the label image is highly regular. The labels of pixels surrounding a particular pixel are very good at predicting what label that pixel should have. (The simplest way this manifests is the fact that a pixel surrounded by pixels with a particular label is almost certain to share that label.) There should be some way to share information from pixels to their neighbors about what class they believe they belong to. [18], for example, solves this problem by using features from throughout a superpixel region to classify the region as a whole, and learns to consider the labels of groups of superpixels to modify the labels on individual superpixels. Semantic Texton Forests use a purely appearance based model, but pooling is done in large squares, effectively reducing the resolution of the output but increasing the accuracy within each larger pixel. Other approaches use Markov Random Field approaches to associate data from neighboring appearance model results.

Visual filters, on the other hand, don't separate out the appearance model from the context model. Instead, both are learned together, in an approach most similar to the efficient auto-context used in [19]. A first filter is trained directly on samples from the training data, and then run to create label maps for all the training data. Then a second filter (called a second layer) is trained using features that sample not only from the training set, but also from the corresponding label maps created in the first step. This is used to create its own label map, which is used to train a third filter, and so on. At each step in this process, the label maps improve in accuracy, until after a very few steps the improvement levels off. This could also be described as a variation on the idea of LeCun's convolutional net, in the way it uses shared weights. (Two differences from the convolutional net architecture are that the higher layers don't reduce in size and there is no backpropagation—instead, each layer gets feedback from the training labels directly, as in auto-context.)

The overall algorithm is as follows:

Training

1. A set of training images are collected.
2. Corresponding label maps are created.
3. For each layer,
4. Features are collected at many random locations within these training pairs.
5. A single-layer (linear) perceptron classifier is trained on these features.
6. For each training image,
7. A feature is collected centered at each pixel in the image.
8. The perceptron is used to label the feature.

9. An estimated label map is created from these labels.

Testing

1. For each test image

2. For each layer,

3. Follow steps 6-9 above

This can be seen as a method for estimating the probability map Y given an image X. The individual pixels of the probability map, y_i, are given by an integration over the joint probabilities of all the other pixels in the map given X:

$$p(y_i|X) = \int p(y_i, y_{j\neq i}|X)dy_{j\neq i} \tag{3.1}$$

At each layer, the classifier for a particular pixel can focus more or less heavily on the probability it was passed from its own location in the previous layer. This means that we can be sure the classifier will converge for the training samples, because it can do no worse than on the previous layer by simply giving that sample full weight and all the other samples zero weight.

3.3 Visual Attention

Visual filters are typically only slightly less accurate than carefully designed, state-of-the-art image parsing techniques. They have certain other advantages, however. Because only a single layer is trained at a time, they are very fast to train compared to other methods (a model with 100,000 training samples and three layers trained on 150 images takes about ten minutes to train using my laptop.) Each

pixel in a layer can be classified in parallel, so on a multi-core system they are fast to parse images (less than a second on a 256 x 256 image.) These factors, and especially their mathematical simplicity, seem in some ways similar to properties of the human visual system.

When we look at the world, we are able to classify many things within the field of view quickly, simultaneously, and effortlessly. Most models of attention assume that when we first look at a scene, the brain pulls out only simple features such as contrast, information density, saturation, creating what is known as a 'saliency map.' These features are thought to provide cues for where to fixate in an image, and that objects are only recognized when they are the center of attention indicated by the direction of gaze.

A few recent works have shown a situation more complicated and interesting. [20] The earlier experiments in this area simply asked participants to look at a scene and describe what they saw. In this case, the location of fixations was predicted pretty well by the low-level features described above. When participants are asked instead to look for a particular item in a natural image, however, the first few fixations were better predicted by the location of the object to be found. This seems to indicate that from the first glance at a scene, before the brain would have time to do anything that requires anything as slow as conscious reasoning or fitting of a complex model, it is already able to classify many objects in a scene correctly and in parallel. Only after this process is completed do we fixate on the object of interest in order to begin these slower and more accurate processes which require a focused attention.

Visual filters provide one explanation about how this nearly instantaneous recognition may be possible. The probability maps generated for particular labels can be considered as saliency maps for particular objects of interest. Because visual filters are able to operate on the features centered at each pixel separately, the process is purely feed-forward and embarrassingly parallel. An estimate based on the time it takes to generate a label for each pixel shows that the speed of a parallel implementation would be heavily dominated by the time it took to load the image to be processed into memory.

3.4 A Biologically Plausible Version

Visual filters are not directly biologically inspired, but the process maps well onto the model of the vision system in the brain as currently understood. The first stages of the visual system can be seen as a way of creating robust features from raw intensity values. We can run the filters using a biologically plausible version of these features as the inputs. The following is a sketch of the current consensus about the process of object recognition in primates.

The data from the eyes is streamed along the ventral visual pathway beginning in the primary visual cortex (V1) and ending in the inferotemporal cortex (IT). This in turn informs the prefrontal cortex, where the information can be used for taking action. The entire process from V1 to IT only takes about 30 ms in humans. [21] (Information about location in the image also begins in V1 but follows a different path. We do not attempt to imitate this behavior in our model.)

The first cells along the pathway, the simple (S1) cells, are similar to local Gabor filters at a particular orientation and scale. Complex (C1) cells integrate the information from a small number of these S1 cells, responding to oriented edges over a wider range of locations and scales. The input of multiple C1 cells, in turn, are used to create more and more complex filters that respond to particular arrangements of multiple edges over larger and larger areas of the image (S2 and C2 cells.) [22] Cells at the end of this process act like radial basis functions, responding strongly to image regions that contain the pattern of interest, and falling off in Gaussian fashion as the similarity between the input patch and the prototype decreases. [23] Up to this point the process is largely feed forward. But within the inferotemporal cortex, these prototypes receive feedback from the prefrontal cortex [24], influencing the interpretation of inputs so that ambiguous areas are resolved into familiar objects through association with the immediate context. For example, a distant brown blob might be interpreted as a shoe if it is found at the bottom of a leg, or as hair if found at the top of a head.

For some cells in the IT cortex, the visual similarity between inputs is less important than semantic similarity. Cells that respond strongly to frontal views of faces, for example, respond partially to profiles of faces, even though their appearance is not similar. [6]

The biologically-inspired version of visual filters follows this natural model closely for the first stages of processing, approximating the action of S1, C1, S2, and C2 cells. (This part of the system uses a variation on the HMAX features described in [25].) Randomly selected 64 x 64 patches of the training images are fed into

this software, and the results are 256 dimensional vectors which encode much of the shape information in the patches in a compact way.

3.5 Experiments

3.5.1 Penn Fudan Pedestrian Database

The Penn-Fudan Database for Pedestrian Detection and Segmentation contains 170 images with pedestrians (mostly college students) on the campuses of the University of Pennsylvania and Fudan University [29]. The official label set contains labels for 345 pedestrians, but many pedestrians, especially those far away or partially occluded, are unlabeled. I supplemented the labels with my own for all the remaining pedestrians, and used labels from this revised set. For this reason, results may not be directly comparable to other scene parsing techniques on this dataset. The revised labels will be made available to other researchers in the future, however. This dataset was a good test of the abilities of the visual filters. There was enough variation to pose a challenge but not too much to be unlearnable. For this reason I decided to use this dataset for an in depth analysis of how various parameters affected learning. I used 150 images for training and 20 for testing.

3.5.2 Number of Training Samples

After reduction using PCA, the vector space of training samples is 50 dimensional. This is much too large a space to map directly. If one were to partition this space like a Rubik's cube, with just three divisions per dimension, and sam-

ple each of those divisions, it would require 3^{50} samples (7×10^{23}) to characterize the space. The regularities in image and label data greatly reduce the number of samples needed to characterize the variation, however.

Five models were trained, holding all other parameters constant, but varying the number of training samples with 100, 1000, 10,000, 100,000, and 1,000,000 samples respectively. ROC curves are shown in figure 3.2. It is clear that the number of samples is an important parameter, and that more samples increases accuracy. The total number of training samples it would be possible to take from this set of images is around 150,000,000, though approaching that number would likely cause overfitting. Training time increases with the number of samples.

Figure 3.1: Effect of number of samples on performance on the most challenging image in the Penn Fudan test set. Top row: 100, 1000, and 10000 samples. Bottom row: 1E5, 1E6, and 1E7 samples. Only at the highest number of samples was the system able to learn a model that could pull out the two figures from the background.

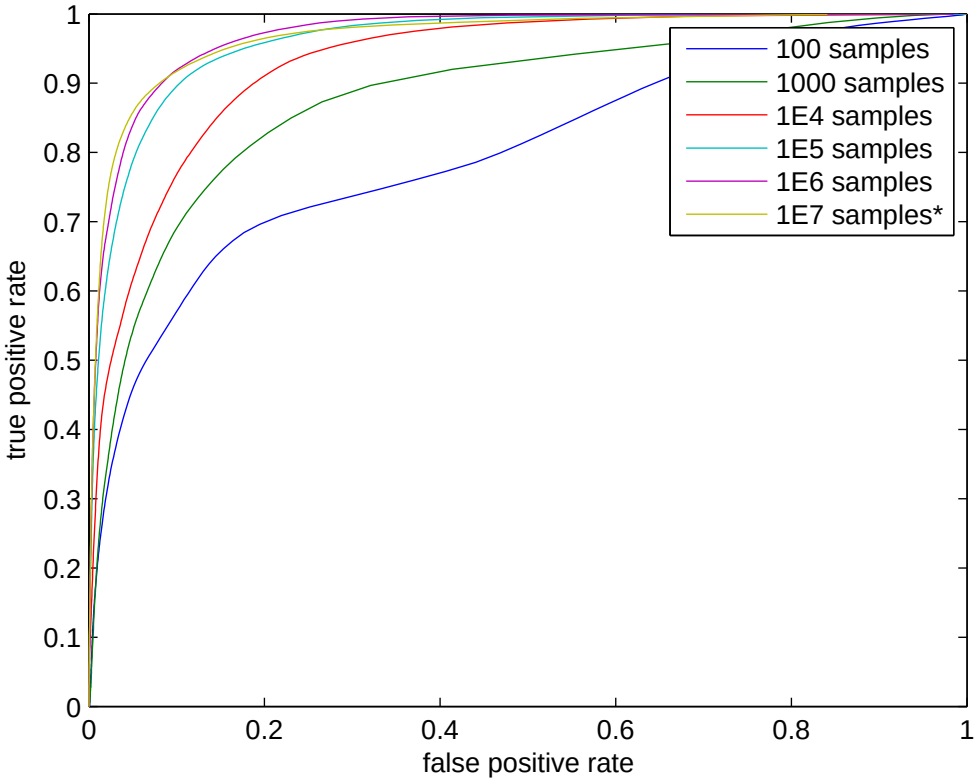


Figure 3.2: ROC curves for models trained on different numbers of samples. Numbers listed are samples per layer. Each model had four layers, and the features were a patch size of 5 x 5 with a 4-level pyramid, compressed to 50 dimensions using PCA. The model for 1E7 samples had 3 layers and a patch size of 9x9. These settings were chosen to attempt to build the best possible model on this dataset.

3.5.3 Patch Size

The features used for these tests are a 4-level pyramid of RGB patches. The size of the patches affects how much context is considered within each sample. Because the features should be centered on the pixel of interest, only odd-size patches are possible. Using 100,000 samples, I investigated patch sizes of 1, 3, 5, 7, 9, and 11. The feature with patch size 11 has 11 x 11 x 4 pyramid levels x 3 colors = 1452 dimensions, but all experiments used PCA to change the number of dimensions of the features to 50 regardless of patch size. The results are in figure 4.1. This shows that increasing patch size increases classification accuracy, but this effect peaks at a patch size of 9 x 9. The largest size patches, at the widest pyramid level have a support of 88 pixels by 88 pixels. Since the images used in this test are typically about 350 pixels high, this includes a sizable fraction of all the context of the image within each patch. When four layers are used, influence can propagate over 352 pixels within the training, which is wide enough that most pixels in the images can potentially influence the label of every other pixel.

At large patch sizes, pedestrians have a kind of probability halo surrounding them. This is because pixels near a pedestrian that look like background are more likely to be false negatives (misclassified as background.) The smaller patch sizes, though, are unable to "see" that they are nearby a pedestrian.

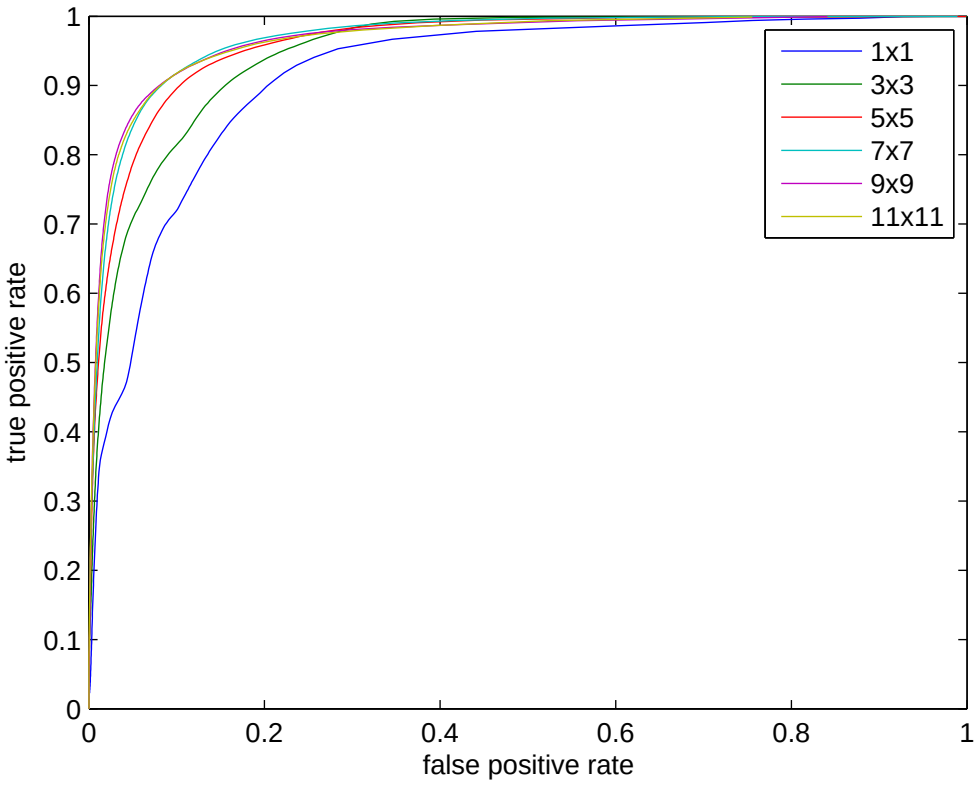


Figure 3.3: ROC curves for models trained on different patch sizes. Performance increases up to a patch size of 9 x 9. Patch size is for a 4-level pyramid.

Figure 3.4: Effect of patch size on performance. Patch sizes are 1x1,3x3,5x5 [top] 7x7,9x9,11x11 [bottom]. Patch size 1x1 uses very little context information, so is mainly determined by the color of the pixel. Patch size 9x9 is optimal on this dataset; 11x11 is similar but begins to ignore local details.

3.5.4 Number of Layers

Each layer allows the filter to take advantage of the more refined probability map created by applying the previous layer to all the training images. This effect quickly stops providing a benefit, however. As seen in figure 3.6, the second layer provides a huge advantage over the first layer, since it is able to take context into account at all. But subsequent layers have little to gain from working with more accurate maps. However, each layer also provides the chance to gather more training samples, and at small numbers of samples this can have a large effect, meaning that using more layers shows improvement beyond the second layer.

Figure 3.5: Output from applying successive layers to test image [left] from Penn Fudan pedestrian dataset. The first layer is confused by bicycles which share the color of clothing and a white shirt that may be confused with white buildings or the sky. But context cues (the presence of the legs and head) allow later levels to erase the noise and fill in uncertain areas.

The results of these experiments showed that the best parameter settings were to use as many samples as is practical, a patch size of 9 x 9, and 3 layers.

3.5.5 Weizmann Horse Dataset

We performed two tests on the Weizmann horse database. [26] This database has large variations in the appearance, lighting, and pose of the horses and variations in background appearance. However, the horses are all seen from roughly the same

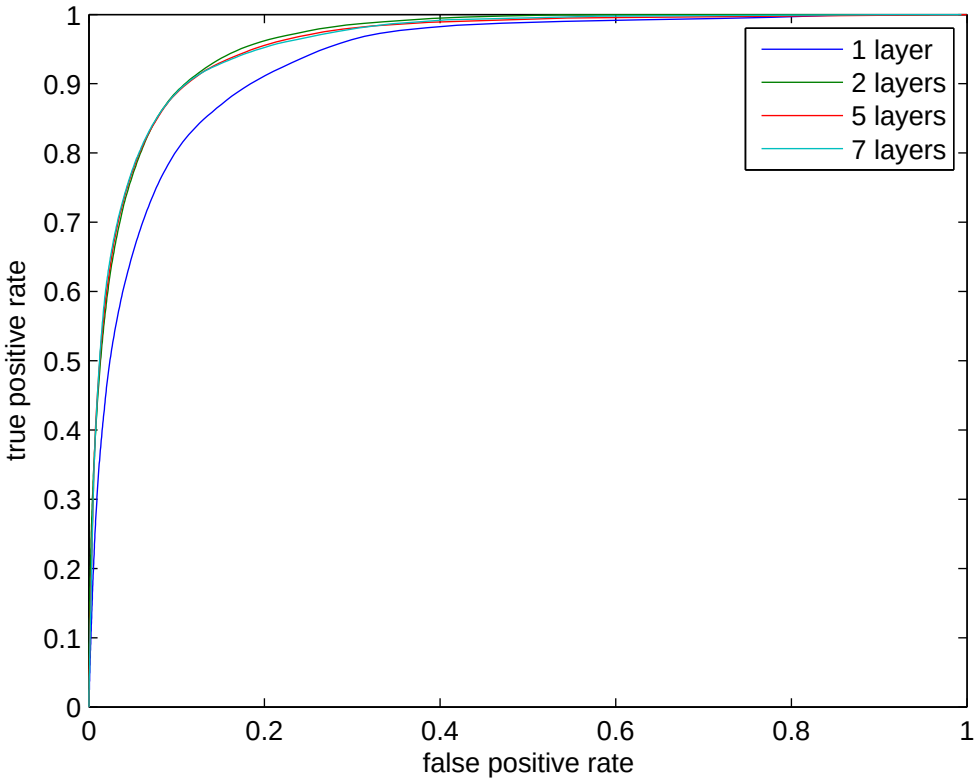


Figure 3.6: ROC curves for models trained with different numbers of layers. The first layer, lacking context, is much lower, but 2 through 7 layers show near identical performance on this dataset.

angle. This means fewer samples are needed to learn the class than would otherwise be the case. Detection is made a little easier by the fact that each image contains only one horse, nearly centered, and there are no partial occlusions. However, due to the windowed nature of the detection algorithm, these factors would not be expected to be very problematic for this system.

For the first test, we used the nonparametric classifier and the biologically inspired features. 500,000 samples were collected at random from the training images for each of the five layers. The system used 64 x 64 patches, and created 256 dimensional samples. In the first case the system was trained on 300 of the images and tested on the remaining 27. The system was able to not only detect the presence of horses, but correctly segment many of the limbs in 24 of the 27 images. For the second test, we used the perceptron classifier and patch pyramid features as above. We used 10,000,000 training samples, a patch size of 9 x 9, and 3 layers. results are shown in figure **??**. Three filters were trained on 218 images and tested on the remaining 109 (as in the other methods compared). The best previously published results are in table 3.5.5; the performance is statistically identical to the state-of-the-art.

Kuettel et. al. [27]	94.7
Bertelli et. al. [28]	94.6
this method	94.5

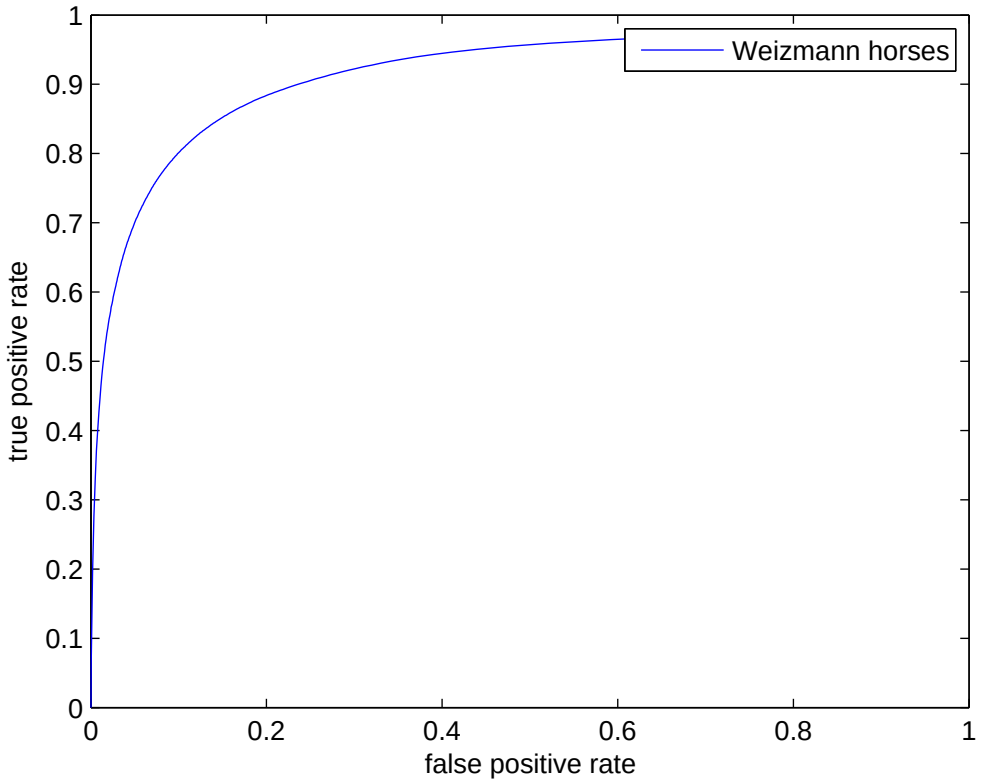


Figure 3.7: ROC curve for Weizmann horse dataset.

Figure 3.8: Output of three successive layers on a sample test image [left] from the Weizmann horse dataset. Each layer removes clutter and shores up confidence of labels based on context from the previous layer.

3.5.6 Poser

In order to create training sets, images must be carefully segmented by hand. The laboriousness of this process limited the size of the training sets available to a few hundred images of a class. Natural images of the class of pedestrians, for example, show variations in lighting, pose, camera angle shape and size, clothing appearance and color, background appearance, and so forth. A much larger training set would seem to be necessary to give examples covering a reasonable amount of the possible variations. In order to create such a training set automatically, we used the commercial Poser software. Poser is a tool for creating high-quality renderings of computer generated humans. Poser has an API for python scripts, which allowed us to create a parameterized library of figures in a variety of poses, lighting situations, camera angles, and clothing styles, in front of a bitmap backdrop showing typical background scenes for pedestrians. At the same time, the label map could be automatically generated. The results were mixed. Using a test set with the same figures used in the training set in different poses, camera angles, lighting situations and backgrounds, the system had 91% pixel labeling accuracy. However, the filters performed poorly on photographs of real pedestrians, with little better than chance levels of accuracy. Apparently the features used were too sensitive to the differences between rendered Poser models and real pedestrians in terms of clothing styles, body shape, and interaction with the background for the filter to be effective. This is a common problem among many machine learning applications: the training set and test set must be drawn from the same distribution, or performance will

be unpredictable. [30] hints at a possibility that using locally adaptive regression kernels as features may be more successful at generalizing from CG puppets to real human figures, given the good generalization between modalities in that paper. This is an area for future exploration.

3.5.7 Limb Detection

The LabelMe dataset provides examples of figures whose limbs are labelled individually. Using this as a training set, we built a limb detector. Selected successful results are shown in figure 3.9. We divided this into a training set of 300 images and a test set of 100 images. Results showed an average of 56% per-class pixel labeling accuracy for non-background classes. The Action Recognition of Static Images [31] dataset does not contain any segmentations. However, it does have the labeled location of limbs as rectangles. Using these as very noisy labels, the system was still able to learn to recognize limbs in a variety of action poses. Results are shown in figure 3.9. In general, those sports whose background was relatively consistent between examples were able to be learned successfully, while other sports had inadequate training data to be learned well by the system. This highlights one real difficulty in using what is essentially a scene classification system for object detection: because the background class is treated equally with the object of interest, the system depends on learning a model of the background as well as of the foreground. If the backgrounds in training are insufficiently similar to those in testing, object recognition will suffer even on identical objects in the two images.

Figure 3.9: Selected images of limb detection on Action Recognition dataset. All training and test images were resized to fit in 300x300 square.

Figure 3.10: Example of the low-quality training labels generated for this dataset.

3.5.8 Semantic Edge Detection

Many features of interest in an image are contour-like rather than area-like. Could visual filters be used to identify these kinds of features? We trained and tested a model on the Berkeley Segmentation Data Set and Benchmarks 500 [32]. This datset contains 300 training images and 200 testing images. In order to make sure that the edges were well-sampled in the training, we thickened them before training to 6 pixels in width. The results on test images can be seen in figure 3.11. A similar experiment was done to test whether the system could learn to recognize the edges between individuals in a crowd. One of the drawbacks of scene classification is that unlike other recognition systems, it cannot differentiate between one large object of a given class and two smaller objects of the same class that happen to be partially occluding one another. The successful detection of semantic edges suggested that it might be possible to learn the edges between occluding objects of the same class as a kind of "outline" class.

We trained a visual filter classifier on a modified labeling of the Penn-Fudan dataset. Three labels were used: pedestrian, background, and pedestrian outline. The outline pixels were the outermost six pixel shell within the original pedestrian label. Sample results are shown in figure 3.12. The system was able to learn to recognize edges that separated the pedestrian class from the background and one pedestrian from another within a large region of occluding pedestrians. Notice, however, that the horizontal portions of the outline (separating the top of the head and bottom of the feet from the background) were not learned. This is likely due

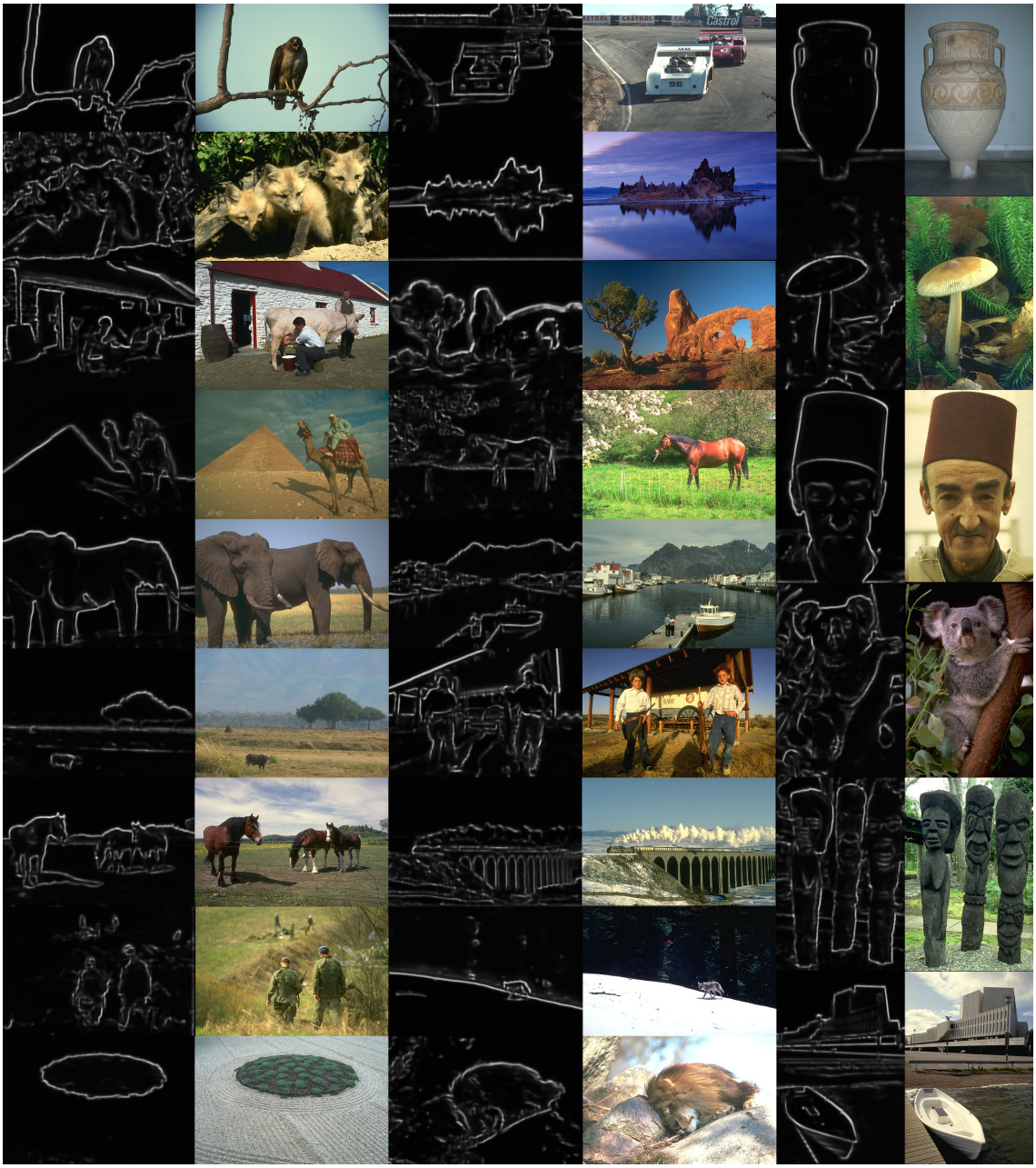


Figure 3.11: A sample of results from a salient contour detection filter trained on the BSDS300 training set and run on the validation set.

to the fact that the total area belonging to these regions was so small (0.16% of the image) that the class was not well sampled, with an average of only 1.3 samples being randomly selected from within a horizontal outline per image. This illustrates another problem with scene classification techniques: classes with small area are easy to lose as the system weights false negatives of a foreground class the same as true positives of the background class, so that it is often less costly to simply take the loss and mislabel the foreground object as background if the foreground object is too small.

3.5.9 Experiments with Different Features and Classifiers

A learned statistical classifier is any function that, based on training pairs, outputs a hypothesized label for any given input. Originally, visual filters used a series of single-layer perceptrons, one for each iteration of the algorithm. We experimented with replacing this with a sophisticated non-parametric classifier that makes use of anisotropic Gaussian kernels. Details of these experiments can be found in [35]

Visual filters can work with any dense features as input. For most tests we used pyramid RGB patch features, which are very fast to calculate. To create a more biologically plausible system, we tried HMAX features [33]. We also experimented with dense SIFT features and dense kernel descriptors. Dense SIFT and HMAX ignore color, which is a strong cue for object class. All three (HMAX, SIFT, and kernel descriptors) lose information at the finest scales due to the size of their

Figure 3.12: Examples of how a learned outline class (white) can separate occluding figures (black). This is only possible with visual filters, not any other scene parsing technique. (Note the incorrect labeling of the gaps between the columns in the left image as legs of a human figure.)

associated kernels. The features used in visual filters are too high dimensional for approximate nearest neighbor algorithms to work well. The 100 nearest neighbors will contain some correct matches but also many incorrect matches. The usual way to weight the neighbors is with a Gaussian function on the distance from the point to be estimated. Unfortunately, in high dimensional spaces, all points are approximately the same distance apart. This is one aspect of the 'curse of dimensionality.' However, the relevant data lies on a lower dimensional manifold embedded in this 256 dimensional space. Because of this, using adaptive anisotropic kernels gives a substantial improvement over the standard isotropic Gaussians. [34]

The advantage can be seen in figure 3.13. Ten points forming an expanding spiral. The points represent features. The spiral is 2-dimensional for illustrative purposes—the actual features are points in a space with fifty or more dimensions. In the first illustration, the weights of each feature are given by an isotropic Gaussian function. When the features are very similar, the points are close together, and the interpolation between them is reasonably accurate. However, when they are widely spaced, each feature lies in its own island. Test features which are very similar to one particular feature will be classified correctly, but ones that lie halfway between two training features will not be.

In the second illustration, anisotropic kernels are used. These are elongated in the direction of neighboring features of the same class. In this case, the features form a nearly connected spiral, correctly estimating the shape of the underlying manifold.

Figure 3.13: anisotropic gaussian kernels [left] and isotropic gaussian kernels [right] on the same ten data points.

3.6 Future Directions: Non-parametric Classifiers

There are some advantages to using a non-parametric classifier over using a trained neural network. First, training time when using large numbers of training samples is much less. What corresponds to training time for non-parametric classifiers is building the tree of nearest neighbors, which only takes a few minutes even on large training sets. This means that adding in new data doesn't require expensive retraining.

Second, because these training features form a set of points, set-theoretic operations such as intersection, union, and difference are easy to accomplish. Given a class (such as 'mammals') and all its specializations ('dogs', 'cats', 'bears' etc...) the classifier is the union of the classifiers for all its specializations. The intersection of two classifiers is a more specific classifier. Taking the intersection of a material classifier for 'metal' and an object classifier for 'toys' yields a classifier for 'metal toys.' Since this classifier has fewer samples than either the material or the object classifier, it is more efficient to run than either of the original classifiers alone.

(Running both classifiers and taking the intersection of the results would take more computation.) Negation and intersection operations would yield detectors for 'non-metal toys,' and 'metal non-toys.' The set of samples for the physical parts of an object divide up the set of samples for the object as a whole. Interaction with a source of world knowledge (such as the Cyc knowledge base) could be used to create new detectors on-the-fly for specific applications.

3.7 Conclusion

Visual filters are a flexible, fast, and simple way to parse images. Their ability to handle contours as well as regions is unique among image parsing techniques. They can be used with a wide range of features and classification methods. They provide insight into how the human visual attention mechanism can work so quickly and adaptably.

Chapter 4: Material Scene Parsing

4.1 Background

Texture classification has been explored in the computer vision community since the 1960s [4], and texture segmentation since 1980 [36]. (For a good overview of the current state of these problems, see [38].) Originally, texture classification was primarily concerned with two-dimensional grayscale textures viewed directly, such as the Brodatz texture dataset [45]. More recent datasets such as the UIUC texture database [46] have included color and textures with some depth under a variety of lighting conditions. In these examples, the depth dimension is very small compared to the other two dimensions, and is directly toward the camera. As such, these approaches are generally useful in medical imaging (with two dimensional slices), distant sensing (where the depth dimension is generally small) and similar problems.

What could be called "material scene parsing" is a more complex challenge. It refers to classifying the physical material composing an object in a scene, labeling the regions as in 'scene parsing' but with strictly material labels. (Scene parsing has always contained a poorly delineated mix of object and material labeling—for example, [16] classifies outdoor scenes with the labels *sky, cloud, grass, foliage, sand, snow, water, rocks, pavement,* and *buildings*; most of which can be considered

broad classes of materials. However, some of the same regions might be labeled as *tree, road,* and *lawn* and be considered as objects in another system.) Materials generally can be thought of as being composed of a collection of textures: both rough cut pine boards and polished mahogany might be labeled a 'wood' texture under a particular classification scheme. The lighting conditions are more general and include cast shadows. The texture is also no longer considered to be camera-facing, but to form part of the surface of objects with unknown geometry.

In 2001, [39] built a system that learned to classify the BRDF (bi-directional reflectance distribution functions) of a shape given a single image under unknown illumination. The system depended, however, on knowing the underlying geometry of the shape, limiting its usefulness. While BRDF is an important quality in how we recognize materials, many materials have a BRDF that varies spatially across a surface—the grain of wood or the dye in a floral print, for example. In 2010, [40] attempted to classify images containing objects of a single material under unknown illumination. This system used a diverse set of features:

- Color: 3x3 RGB pixel patches
- Texture: oriented multi-scale Gabor features and SIFT features
- Microtexture: texture features (as above) on the difference between the original image and a bilateral filtered version, to capture surface details
- Shape: curvature of Canny edges at multiple scales
- Reflectance: HOG features tangent to and normal to curves in the image

These features were quantized into visual words and LDA was used to separate the classes. The system had an overall recognition accuracy of 44.6% on the Flickr

Material Database [47].

In response to this, [41] showed improved performance of 54% accuracy on the same dataset in 2011. The features used in this classification were Kernel Descriptors (described in detail below.)

These two papers addressed material classification, but did not attempt to apply their techniques to the related problem of material scene parsing. All of the images used to test these techniques consisted of a single material, or a single material against a masked off background. To create a system that could label materials in a diverse scene, I re-implemented and extended the more accurate of these two methods by segmenting the scene into superpixels and applying the classification technique they described to each superpixel separately.

There is currently no publicly available dataset of images labeled by their component materials. I hand-labeled seventeen images for testing purposes, and evaluated these using leave-one-out training. To increase the breadth of coverage of the dataset, I also wrote a program to automatically download up to one hundred texture images of any label from Google Images, and apply perspective and intensity transformations to these to simulate the effects of viewing them from a different point of view. This makes the assumption that the textures are essentially flat and diffuse, an assumption that is only warranted for some classes of textures but is a reasonable assumption for certain viewing angles for isotropic materials.

In principle visual filters could also be used for materials, but this has two difficulties. First, it would require creating a large training set from real images. Second, the filter necessarily takes into account shape information, so a wooden

horse or plastic horse would be more likely to be labeled as being covered in the material 'hide' than it should due to this bias. For these reasons, I decided to take another approach that is not influenced by shape, and can be trained using textures downloaded from image searches.

4.2 Algorithm Outline

1. The user enters a list of terms to train on.
2. The system collects examples of labeled textures by doing an internet image search for the terms listed, rejecting images which are not textures but some other kind of image. (These training images may be supplemented with training images from other sources as well.)
3. These texture images are warped to allow for more robustness to perspective transformations.
4. The saturation and intensity of samples are varied to increase robustness to lighting changes and camera differences.
5. The system gathers kernel descriptors and color histograms from these textures.
6. The dimensionality of these descriptors is reduced using the Large Margin Nearest Neighbor technique. (This avoids the issue of PCA throwing away information that may be useful for classification.)
7. A nearest neighbor, SVM, or multi-layer perceptron classifier is trained on these features.

8. Test images are segmented using entropy rate superpixel segmentation.
9. Features are gathered from each superpixel and classified using the classifier from step 7.
10. A map is generated for the entire image for each label, showing the energy of that label (a stand-in for probability) for each superpixel.

4.3 Image Search

The exact algorithm behind Google Image Search is a trade secret, but it appears to use explicit or implicit textual metadata to make its selections rather than image features. A search that includes the word "texture" will mainly return results that have been created for visual artists making textured 3-D models.

The color histograms and kernel descriptors from a texture image will tend to be fairly similar to other features from the same image. Other types of image that show up in the search will have very different features from one part of the image to another. This was used to eliminate non-texture images from the training samples.

4.4 Kernel Descriptors

Orientation histograms, such as SIFT and HoG, have proven to be very useful features for many computer vision algorithms, due to their robustness in the face of moderate levels of 3-D rotation, lighting, or geometry change. While they are undoubtedly useful, their design seems to be somewhat arbitrary. Why the particular binning used in SIFT, rather than some alternative arrangement? While similarity

of SIFT features captures something about similarity of shape, gradient magnitude, and gradient orientation, it is difficult to say precisely what. Kernel descriptors [42] are an attempt to capture what is useful about SIFT and HoG but to do so in a disciplined way. The gradient match kernel, K_{grad}, is defined as follows:

$$K_{\text{grad}}(P,\ Q) = \sum_{z \in P} \sum_{z' \in Q} \overline{m}(z)\overline{m}(z') k_\theta(\overline{\theta}(z), \overline{\theta}(z')) k_p(z,\ z')$$

where P and Q are patches from two different images, $\overline{m}$ is the normalized gradient magnitude, k_p is a Gaussian position kernel, and k_θ is a Gaussian kernel over orientations (using $\overline{\theta}(z) = [\sin(\theta(z)) \cos(\theta(z))]$ to maintain rotational symmetry in the distance function). Thus, the kernel is composed of three parts. The first part captures the normalized gradient magnitudes at each point in the patch. The second computes gradient orientation such that similar kernels will have similar orientations in a rotationally anisotropic way. The third part takes into account the position of each point within the patch.

Using this simple way of looking at orientation histograms as kernels, we can extend the principle to include other image attributes. A color match kernel

$$K_{\text{col}}(P,\ Q) = \sum_{z \in P} \sum_{z \in Q} k_c(c(z),\ c(z')) k_p(z,\ z')$$

combines a position kernel and a color-distance kernel in the RGB color space. A shape descriptor matches local variations in intensity in a similar way, matching features like LBPs (Local Binary Patterns from [37]).

The main drawback to using kernel descriptors directly is their high dimensionality. In the material classification paper, they suggest using Large Margin Nearest Neighbor on the kernel descriptors for supervised dimensionality reduction. They treat each image as the single feature vector formed by taking the arithmetic mean of the features from all of its patches. These reduced dimensionality vectors can then be classified using any machine learning method.

4.5 3-D Color Histograms

Because color plays such an important role in discriminating materials, it makes sense to treat distance between color features in a precise way. The 3D color histogram in Lab color space can capture the color distribution accurately, if there are enough bins. The distance between two histograms is properly measured using the Earth Mover's distance metric [48], so-called because each pixel can be thought of as a bit of earth in one bin of the Lab color cube, and the similarity between two histograms is the amount of work it would take to move dirt from bins to other bins to match the histogram in a second image. EMD tends to be expensive to compute, but if there is a threshold beyond which all distances are the same, the calculation time becomes small [49]. In addition, comparing the same histogram to many different histograms also speeds up the algorithm. I used the FastEMD code from [50].

4.6 Superpixel Segmentation

Superpixels [43] group together pixels which are local and coherent. A segmentation of an image should ideally be possible by selecting a set of superpixels for each segment; the breaks between superpixels should correspond to the breaks between segments. Each superpixel should contain only a single material, so assigning a single label to the entire superpixel is appropriate. One of the nice features of superpixels is that one can gather multiple color or texture features from within a superpixel without worrying that more than one texture is being sampled. With entropy rate superpixels [44], University of Maryland researchers introduced a novel objective function for superpixel segmentation, combining the entropy rate of a random walk on a graph and a balancing term to encourage size consistency between superpixels. Using an efficient greedy algorithm for graph decomposition, entropy rate superpixel segmentation outperforms other superpixel algorithms on the standard Berkeley segmentation benchmarks quickly and effectively.

4.7 Experiments

4.7.1 Experiment 1: Flickr Material Database

The first experiment was conducted on the Flickr Material Database [47]. Although segmentation is useless on this dataset (since each image contains only one material) this was the only direct comparison possible with other techniques. The results are shown in 4.1. The inclusion of color histogram features allows it to

show slightly higher performance on the dataset than the kernel descriptors method it was based on. This was mainly included as a sanity check to make sure I had implemented the kernel descriptor method correctly.

Table 4.1: Comparison with other material classification techniques.

Kernel descriptors + color histograms	56%
Kernel descriptors	54%
MIT material classification	45%

Figure 4.1: Example images from Flickr Material Database.

4.7.2 Experiment 2: Hand-Labeled Imagery

I labeled 17 images with the following labels: Grass, Wood, Metal, Sand, Plastic, Stone, Foliage, Sky and Unknown. These images were chosen to have large

contiguous regions of the same material, so performance on these images may not reflect real-world performance on arbitrary images.

Table 4.2: material labeling accuracy

per-pixel accuracy	65%
per-superpixel accuracy	68%

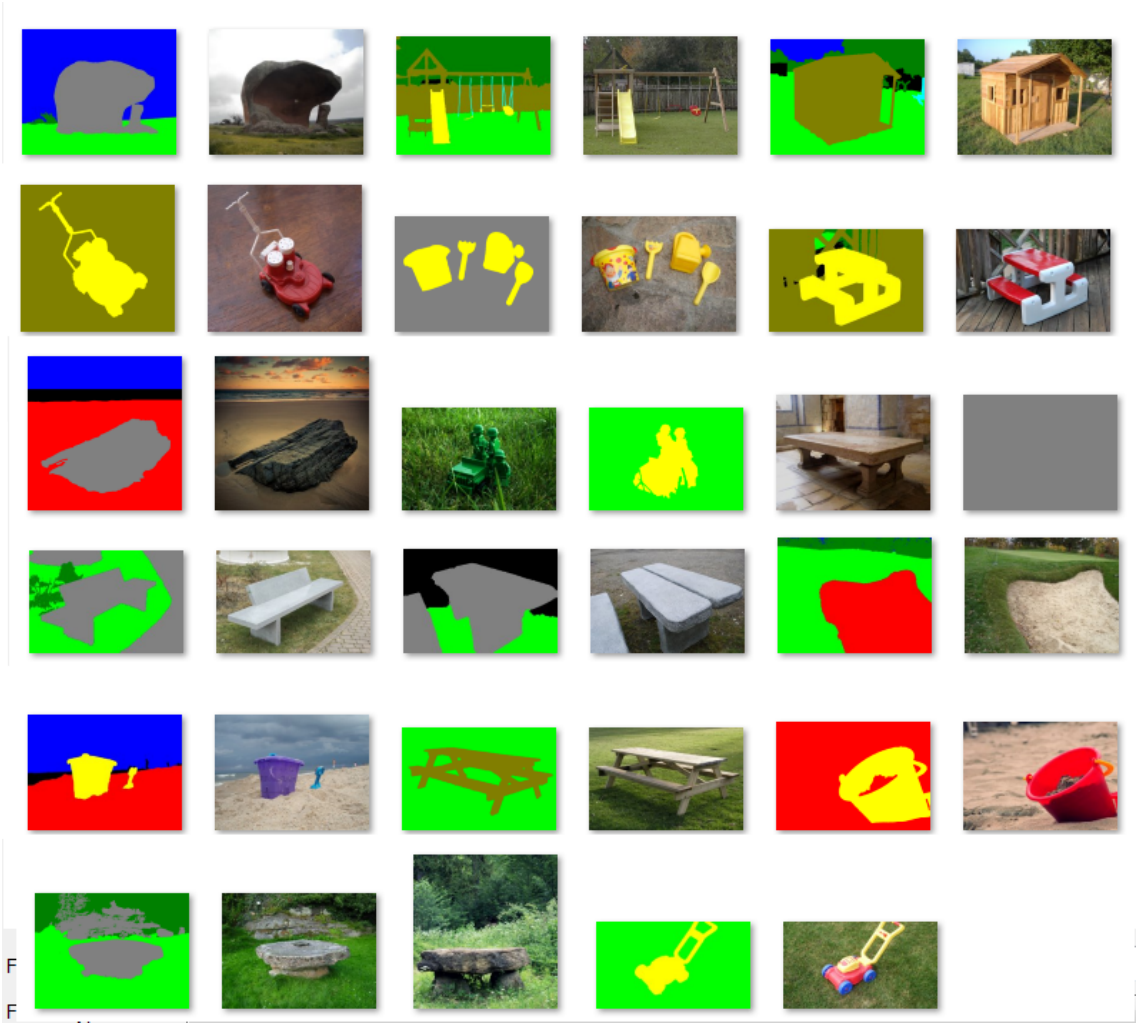


Figure 4.2: Images from test set.

These are somewhat higher than the results on the Flickr material dataset, mainly because it avoids troublesome classes like glass.

4.8 Discussion

Material is just one of many attributes that could be looked for in images. Others often found include shape, color, and size. In more limited contexts we could look for other attributes (expressions of people, for example, or detecting the age of a book from its typeface.) Each of these attributes requires a separate set of features. Objects are recognized by recognizing several attributes that typically belong to the object. If we were to try to recognize a jaguar, for instance, we would look for a large, fur-covered, black-spotted orange cat-shape.

The Cyc knowledge base contains information on several of these attributes for some objects, though it is far from complete in this regard. By making use of knowledge about what materials objects are primarily composed of, we can improve our estimate of object segmentation by using material segmentation, and vice-versa. (This is explored in chapter 6.) Including more attributes could further improve detection results.

In a way, material and other descriptive attributes can be thought of as a feature, like SIFT, HoG, or kernel descriptors but at an object level.

The accuracy of material classifiers is low, but little effort has been put into creating adequate training samples or finding what features work the best yet. As machine learning methods improve, the accuracy will likely improve significantly.

Chapter 5: Action Grammars

5.1 Introduction

How do humans come to understand, recognize, and replicate actions? Even if we have witnessed several occurrences of the same activity, each will be unique in terms of the order actions are performed, the explicit motion of the limbs involved, and the appearance of the objects involved. Somehow, the sensory data must be stored in a greatly compressed representation that captures relevant information while discarding what is irrelevant. This representation must be capable of handling actions of any complexity, where activities are composed of previously known actions and sub-actions.[1]

This suggests that the brain uses a similar method for understanding both language and actions. This idea has support on neuroscientific, anthropological, and behavioral grounds. In the early 1950's, psychologist Karl Lashey suggested that syntax may apply to goal-directed actions as well as to language [87]. Archaeologist Leroi-Gourhan argued that tool making and use reflects a capability for compo-

[1]Much of the text in this chapter is adapted from the paper 'Using a Minimal Action Grammar for Activity Understanding in the Real World' published in IROS 2012. This paper had the following co-authors: myself, Ching L. Teo, Yezhou Yang, Cornelia Fermüller and Yiannis Aloimonos. My contributions include the design and implementation of the two practical action grammars described below.

sitionality of structures, linking language and action. Two-year old children have been shown to have the ability to recognize and reproduce hierarchically organized actions [63, 96]. Additionally, the same parts of the brain that have long been understood to be used in language production (such as Broca's Area) have been found to be crucial to the process of action planning [68, 69].

If such a representation is taken to be an innate, central aspect of both language and activity understanding, it must be simpler and more fundamental than the grammars we learn for each individual language. It also must be a generative grammar (rather than one used purely for recognition) in order to allow an individual to learn to perform actions by example. Chomsky's minimalist program is an attempt to discover such a universal generative grammar for language, with the expectation that it plays a more general cognitive role. A generative grammar consists of a set of elements and a set of production rules that allow the formation of grammatical sentences. Context-free grammars are generative grammars which have recursive rules which allow nesting of elements in the same type of elements. Although context-free grammars are sufficiently expressive to handle the complexity of language, they cannot account for what we actually see in natural languages, such as agreement (in case, number, or gender) and reference (such as relative clauses.) These long-distance dependencies cannot be captured by context-free grammars. The Chomskyan Minimalist Program deals with this through a number of transformations on the output of context-free grammars [66].

In [85], Pastra and Aloimonos introduce a minimalist grammar of action which defines the set of terminals, features, non-terminals and production rules for such

a grammar in the sensorimotor domain. However, this was a purely theoretical description. The action grammars used in our experiments are an implementation of such a grammar in a system that is capable of sensing and interpreting real-world situations under a wide range of natural conditions. Such a representation is a natural summary of the important aspects of an activity, which abstracts away such details as who is performing the action, where it is being performed, how the objects involved are spatially located, and the appearance of the objects and body parts. What is left is a tree structure that captures the order in which tools, objects and hands are brought together and separated, a structure which is easy to store, search, and compare to other such trees.

In order to make the use of the grammar practical for real robots and surveillance, we have not merely created a demonstration, but designed the system so that it will be able to handle and abstract away a wide range of realistic conditions, such as varying viewpoint, lighting, surrounding environment, object and actor appearance.

The activities we are attempting to recognize and understand are complex, concrete human activities. Actions like "stirring" or "tightening a bolt," the traditional purview of action recognition techniques, are represented by a single node in the action tree. (For this reason, we refer to what we are doing as "activity recognition" rather than "action recognition.") Abstract actions, like "doing research," or "playing soccer" contain important steps which take place in terms of mental or data structures, which we have no way to detect or estimate with the current setup. Instead we are looking at multi-step activities which involve the manipulation of

physical objects towards some goal state. This is basically any form of manual labor: the physical work of craftsmen, home builders, factory workers, chefs, janitors, and so forth. These are also largely the kinds of activities which we would hope for a general purpose robot to be able to perform.

"Action recognition" interacts with activity recognition in two important ways, both assisting with and being assisted by activity recognition. First, activity recognition provides important context for action recognition. One of the main difficulties in action recognition is finding when the action begins and ends. Forming an action tree provides natural endpoints for individual actions: these actions occur between the time a tool (including the hands) comes into contact with an object and the time when it breaks such contact. Knowing what the tool and object are provides significant constraints on what the action might be, reducing it to a handful of possibilities with any significant probability of occurring. Second, when action recognition is performed, the action can be used as a label on part of the activity tree, which improves our ability to match with similar activities.

5.2 Recent Works

The problem of action recognition and human activity has been an active research area in Computer Vision, motivated by several promising applications, such as human-computer interface, video indexing and retrieval and video surveillance, etc. Several excellent surveys on the topic of visual recognition are available [82, 93]. But non-visual descriptions, using motion capture systems, have also been of

interest in Computer Vision and Graphics. Many of those studies are concerned with dimensionality reduction techniques, that provide a good characterization for classification [64,80,95]. Most of the focus in visual action analysis was on the study of human actions that were characterized by movement and change of posture, such as walking, running, jumping etc. The dominant approaches to the recognition of single actions compute statistics of spatio-temporal interest points [67, 78, 97] and flow in video volumes as descriptors, or represent short actions by stacks of silhouettes [71,98]. Approaches to more complex, longer actions employ parametric approaches, such as Hidden Markov Models [75], Linear Dynamical Systems [89] or Non-linear Dynamical Systems [65], which are defined on tracked features or optic flow presentations.

To capture the semantics of complex activities, higher level reasoning methods are required. A number of approaches use stochastic context free grammars with the primitives beings body parts [88] or trajectories [74], and some also include the interaction with objects [83]. To model the temporal constraints, several approaches have used Hidden Markov Models, which exploit the relation between specific objects and actions [73, 84]. A related class of approaches use dynamic Bayesian networks to divide the temporal sequence into sub-sequence and define relative temporal relations [70, 79, 86].

Most closely related to our work are a few recent studies on hand manipulation actions. In [94] manipulation actions are represented as a sequences of motion primitives. The process is modeled using a combination of discriminative support vector machines and generative hidden Markov models. In [76] hands and objects

segmented from the video and shape-based hand/object features and manipulation features are defined to provide a sequence of interrelated manipulations and object features. Semantic manipulation object dependencies are extracted using conditional random fields. In [92] manipulations in a breakfast scenario are analyzed. The image sequence is represented by an activity graph that codes spatiotemporal object interactions. Event classes are extracted from the activity graphs, where each event class encodes a similar pattern of spatiotemporal relations between corresponding objects, but the objects are known beforehand. While all these approaches use task-dependent primitives, our approach is general; its basics are simply the merging and parting of objects. A similar idea was pursued by [62] for the analysis of short stereo video sequences of of hand motions manipulating a number of objects. Relations between object at decisive time points during manipulation, such as when two objects touch or overlap, are stored in a transition matrix. Using simple sub-string search algorithms different matrices are compared for recognition. The objects in these sequences are however easily visually recognized, and the approach was only applied to short activities, such as putting two objects on a plate.

5.3 Approach

We describe the overall approach of using the action grammar for activity understanding (see Fig. 5.1) by first introducing the experimental dataset in sec. 5.3.1. Next, we define the action grammar and how it is created in sec. 5.3.2. We then detail how the important subcomponents: hand state determination and object

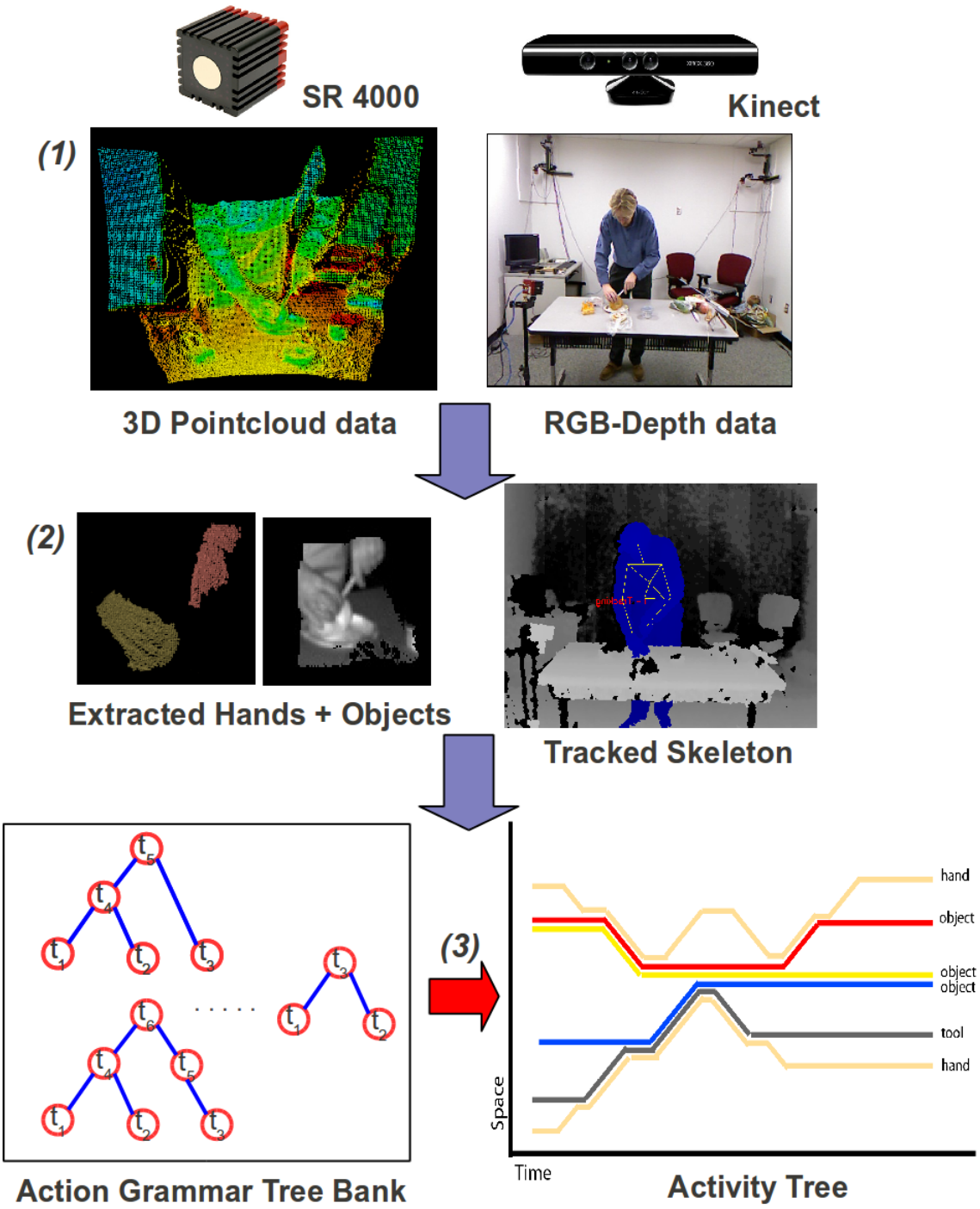


Figure 5.1: Overview of the approach: (1) Pointcloud data and RGB-Depth data are extracted and processed from the SR4000 and Kinect cameras respectively. (2) Hands and objects are detected from pointclouds and the human pose is extracted from Kinect. (3) The detected objects are then combined using an action grammar treebank to produce an activity tree.

recognition are achieved in secs. 5.3.3 and 5.3.4 respectively. With these detections, we illustrate how an *activity* tree can be built (sec. 5.3.5) and be used for comparing the similarity between different trees (sec. **??**) and how the action grammar is useful for separating complex interleaved activities in sec. 5.4.1.

5.3.1 Kinect+SR4000 Complex Activity Dataset

We introduce a novel dataset that contains 5 complex hand manipulation activities performed by 4 different human actors. Each *activity* is defined by the completion of a complex object or entity: for example, making a sandwich. The task of creating this entity is further comprised of 9 specific *actions* which may involve the use of different kinds of hand-tools and objects. Different actions could be concatenated to form novel activities: Cooking vegetables + making a sandwich.

Other well known datasets such as the KTH, Weizmann or Human-EVA datasets [71,90,91] do not involve hand-tools. The human-object interaction dataset by Gupta et al. [72] has only 4 objects with extremely simple actions. The dataset by Messing et al. [81] has only 4 simple actions with tool use. The CMU Kitchen Dataset [77] has several actions performed by 18 subjects for 5 recipes, but many of the actions are blocked from view due to the placements of the 4 static cameras.

The Complex Activity Dataset extends beyond these datasets by considering the compositional aspect of the activity in terms of the entities created by the actions involved in each step. The activities are classified into two general categories: *Kitchen* and *Crafts*, each with 8 separate video sequences captured from

two externally synced and calibrated active sensors: 1) the Kinect which provides RGB-Depth and 2) a Swissranger SR4000 Time of Flight camera which provides Intensity(Grayscale)-Depth. The Kinect camera is positioned frontal-parallel at a distance of $\approx$ 6m from the actor so that we can track the entire body motion, while the SR4000 is positioned $\approx$ 1.5m from the actor on the side so that hand-actions and objects can be clearly seen (see Fig. 5.2). In total, there are 16 video sequences made from different combinations of activities and objects. The sequences are fully annotated with of the names of relevant objects and manipulative actions for evaluation and training purposes. The list of activities, actions and objects considered are summarized in Table 5.1. Sample sequences from the dataset are available in the supplementary material[2].

Activity Class	Name	Actions	Objects/tools
Kitchen	Cooking Vegetables	{*slice, cut*}	{*cucumbers, carrots, tomatoes, apple, chopper*}
	Making Sandwich	{*spread, slice*}	{*bagel, ham, cheese, knife*}
Crafts	Sewing a Toy	{*cut, pin, thread, sew*}	{*cloth, paper, needle, thread, scissors*}
	Card Making	{*cut, paste, write, fold*}	{*paper, glue, scissors, marker*}
	Assemble a Machine	{*screw, push, twist*}	{*wrench, nut, bolt, frames*}

Table 5.1: List of manipulation activities considered.

5.3.2 The Action Grammar

The first grammar has one simple rule, which is applied repeatedly:

[2]More information on the dataset and how the data is collected can be found at: http://www.umiacs.umd.edu/research/POETICON/umd_complex_activities/

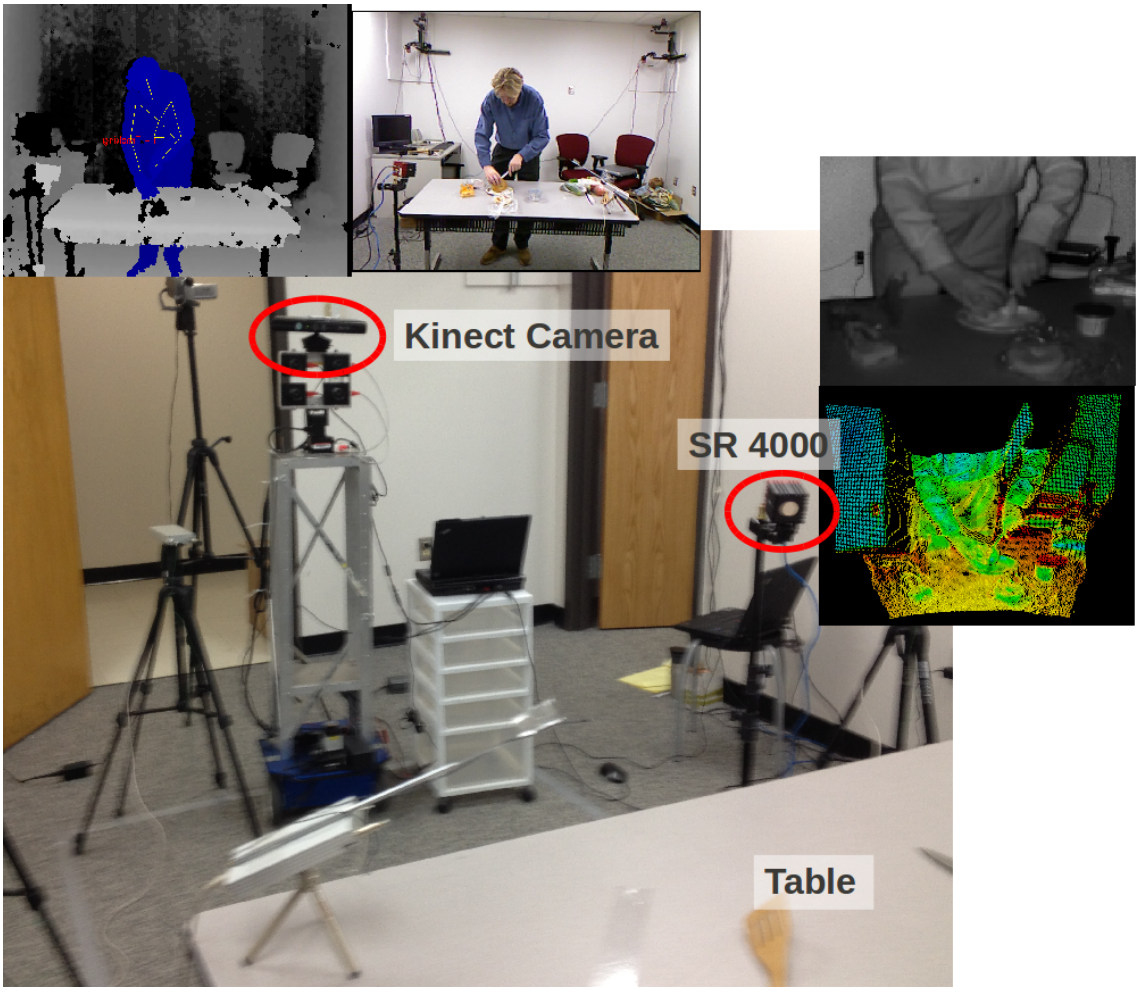


Figure 5.2: Data collection setup. The Kinect is mounted on an Erratic mobile robot base, the SR4000 is mounted on the side nearer to the table where the actions are performed.

An activity consists of

1. Using a tool to bring two objects together, resulting in a new object or a tool

   or

2. Using a tool together with a single object, resulting in a transformed object or tool

This grammar is only appropriate for assembly-type activities where all objects persist throughout the scene. If objects can be created or destroyed (including when they enter or leave the scene) and if objects are also disassembled, this can be handled with a slightly more complex grammar: An activity consists of

1. Using a tool to bring two objects together, resulting in a new object or a tool
2. Using a tool together with a single object, resulting in a transformed object or tool
3. Using a tool to separate two objects. The activity of separating the objects becomes a parent to the activity of handling each of the objects.
4. When an object is used in activity A and reused in activity B, a new activity is formed with A and B as sub-activities.

These tools or objects can themselves be the result of an activity, which gives rise to the tree structure of activities. Hands can be thought of as tools which are not made of other objects or tools. The new "object" can be something like "a piece of bread on a plate" formed by bringing together a slice of bread and a plate. The point is that after they have been brought together, they are treated as one combined object, temporarily, as the bread moves together with the plate.

Realistically, there are certain 'activities' under this definition which are best

ignored. When an object is handled, it may be passed from hand to hand, set down on the table to get a better grip, accidentally dropped, and so forth. These each technically result in a split and merge activity with the hand, but such actions happen at frequent, unpredictable times, and give little insight into the structure of the activity. For this reason, we only keep track of merges between objects and tools, and group together into a terminal (leaf) node 'handling an object' any number of grasping, shoving, dropping, and regrasping actions. Merges with the table surface and floor are ignored. Under some circumstances this would be inappropriate (bouncing a ball or setting a table, for example) but for most of the activities we are interested in it reduces noise and captures the relevant activities.

There are two ways to use the grammar. In this paper we parse the actions that take place, starting with recognizing simple actions (of type 1 or 2, above) and building them up into an activity tree, an example is shown in Fig. 5.6 (right). Every non-terminal node of this tree is an action. The other way to use the grammar would be in a generative way: starting with an activity one wanted to perform, and working out how to do it. One would simply look for an activity one has observed resulting in the final object one wanted to have, find out what the input objects and actions to attain that are needed, and what activities result in *those* objects, and so on, breaking it down to the point that the objects and tools needed are the ones available.

5.3.3 Extracting Hand Locations from 3D Pointclouds

Since we are concerned with *manipulative* actions, the terminals in the action grammar trees are the objects/tools that are currently manipulated by the hands. An approach that passively searches for objects and tools in the video will not be sufficient as many objects are visible on the table but are not participating in the activity. Instead, we actively search for hands in each frame, and determine if the hand is currently occupied with an object or free directly from 3D pointclouds—a binary hand state $H_s = \{occ, free\}$. Once we know the approximate location of each hand and its state, a trained object classifier can then be used only on these regions, which reduces processing time and false positives. Note that we process pointclouds from the SR4000 since it is nearer to the actor than the Kinect and provides a clearer view for object recognition.

The procedure is summarized on Fig. 5.3. The inputs are the pointclouds obtained from the SR4000 and the tracked skeleton from the Kinect[3]. Since both cameras are calibrated, the approximate 3D locations of the tracked hands are known in the SR4000 camera coordinates. However, relying solely on the Kinect for hand tracking is unreliable since it may fail, especially when half the body is blocked by a table (see Fig. 5.2). Our proposed approach is to 1) robustly extract potential hand regions from SR4000 pointclouds, and 2) combine it with the predicted locations from Kinect, so as to determine the final locations of the hands in SR4000 and its hand state: occupied or free. We use PCL 1.4[4] as the main pointcloud processing li-

[3]PrimeSense OpenNI implementation was used to obtain the skeleton.
[4]http://www.pointclouds.org

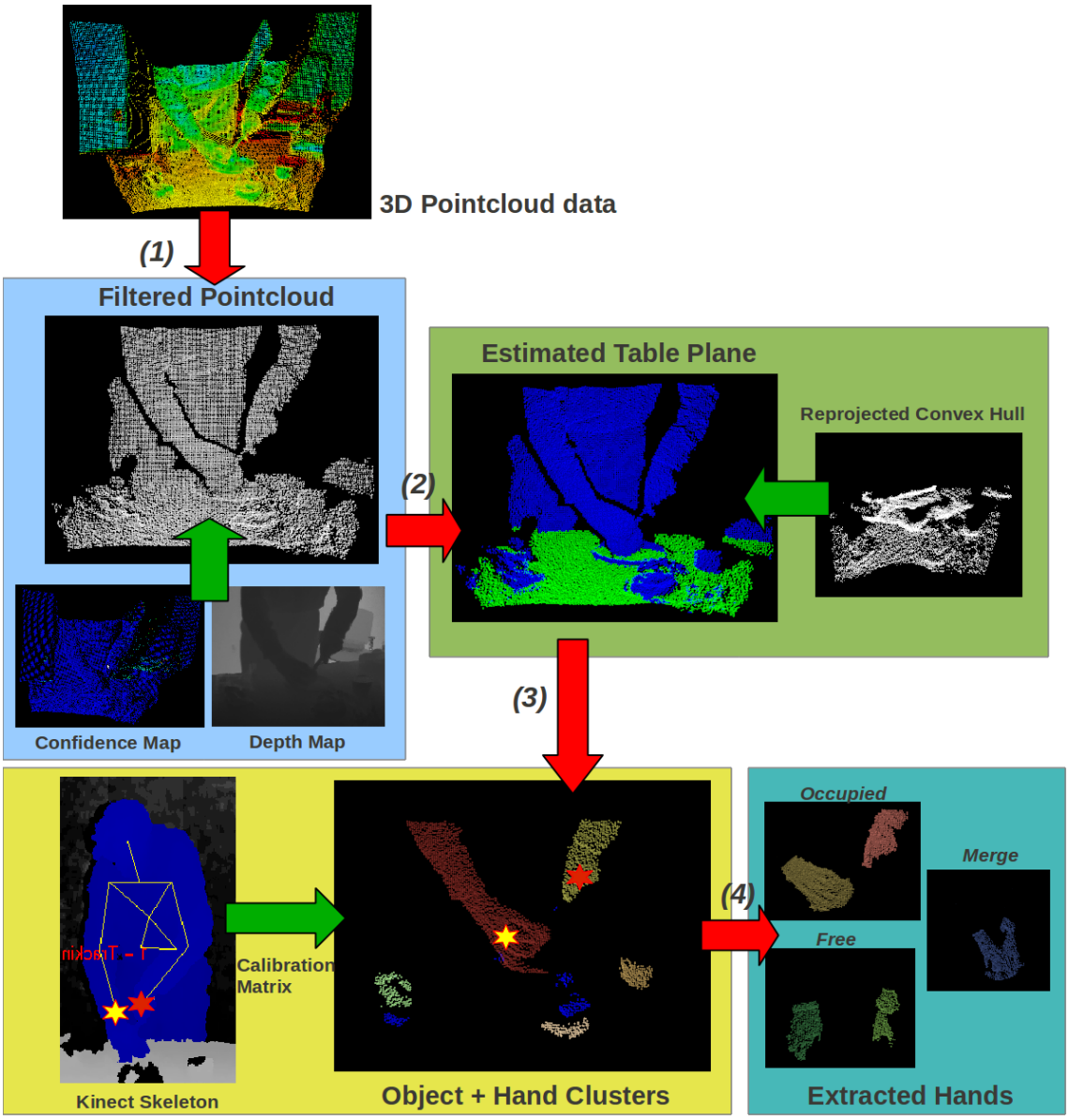


Figure 5.3: Detecting hand locations from SR4000 pointclouds. (1) Outliers are first removed, (2) Table surface is then estimated, (3) Objects and hands are extruded and clustered from reprojected convex hull, and (4) Predicted hand pointcloud locations.

brary to first remove obvious outliers by filtering out points that have low confidence values or those does not belong to any obvious cluster. A plane estimation procedure using RANSAC is then applied to estimate a planar model that represents the table surface. The estimated coefficients are then used to reproject the remaining points so that a convex hull is created from which points in the original cloud that are within the convex hull (table points) are removed. A 3D Euclidean clustering is then applied to obtain reasonable pointcloud clusters. The predicted hand locations from Kinect are then used to extract the hand pointclouds if the location is within a fixed distance threshold of the cluster's centroid. Finally, the extrema of each hand pointcloud is computed from which we use a region growing segmentation algorithm using nearest neighbors to extract the hand and any associated objects/tools that are in contact with the hand (Fig. 5.3(d)). The current hand state H_s is obtained from the difference in the pointcloud sizes against a running average of previous pointcloud sizes. A significant deviation beyond a ratio threshold will indicate that the hand is occupied (ratio > 1) or empty (ratio < 1). In addition, if only a single hand pointcloud is detected, and its current size is approximately equal to the combine sizes of the the left and right hand in previous frames, a *merge* event is raised. This will be important for building the activity tree (sec. 5.3.5).

We later developed a method that used the Kinect data exclusively. The centroids of each 3D cluster are tracked using a Kalman filter. Estimated 2-D location for each object in each frame is computed using dense optical flow from the previous frame, and the 3D cluster at that 2D location is presumed to correspond. This is supplemented by object detection on the RGB image. The tracking algorithm

provides for the possibility that tracks may merge or split, so these events are parsed using the grammar described above. (See figure 5.4.)

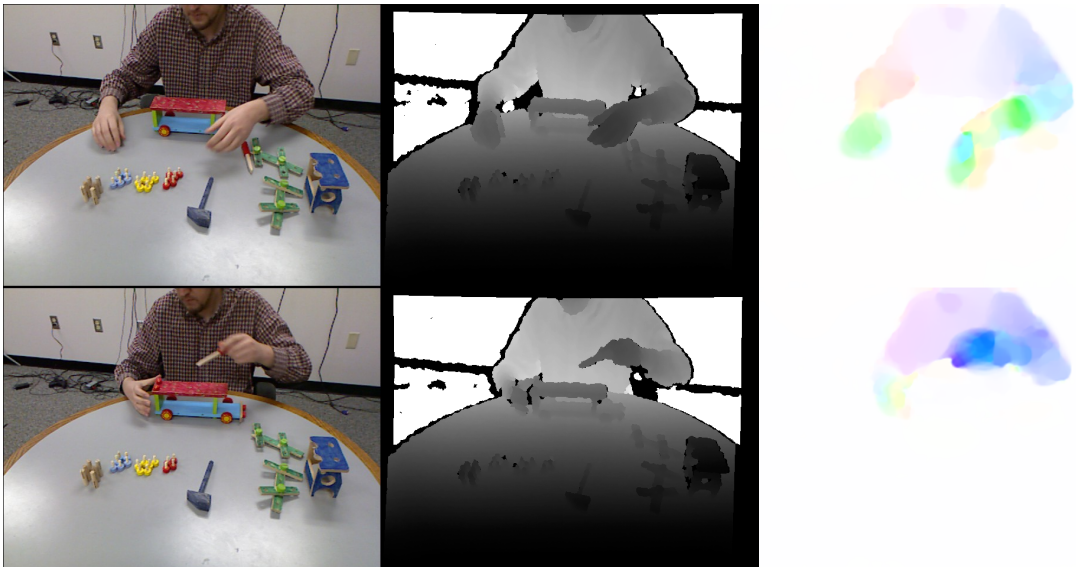


Figure 5.4: [left] Kinect RGB image. [center] Kinect depth image. [right] Dense optical flow computed between this previous frame and this frame.

5.3.4 Object Recognition

The activity tree is built when there are changes H_s for each hand (left and right), and the terminals are the objects (if any) on each hand when H_s changes. Using the results of the predicted hand locations and states described in the previous section, we crop out a rectangular region slightly larger than the hand point cloud size to obtain an intensity image of the potential objects/tools should $H_s = occ$ (Fig. 5.5). We also extract the cropped region whenever a merge event starts or ends.

We extract Histogram of Gradient (HoG) features from the cropped image from which an object classifier is then used to predict the object label. Classifiers

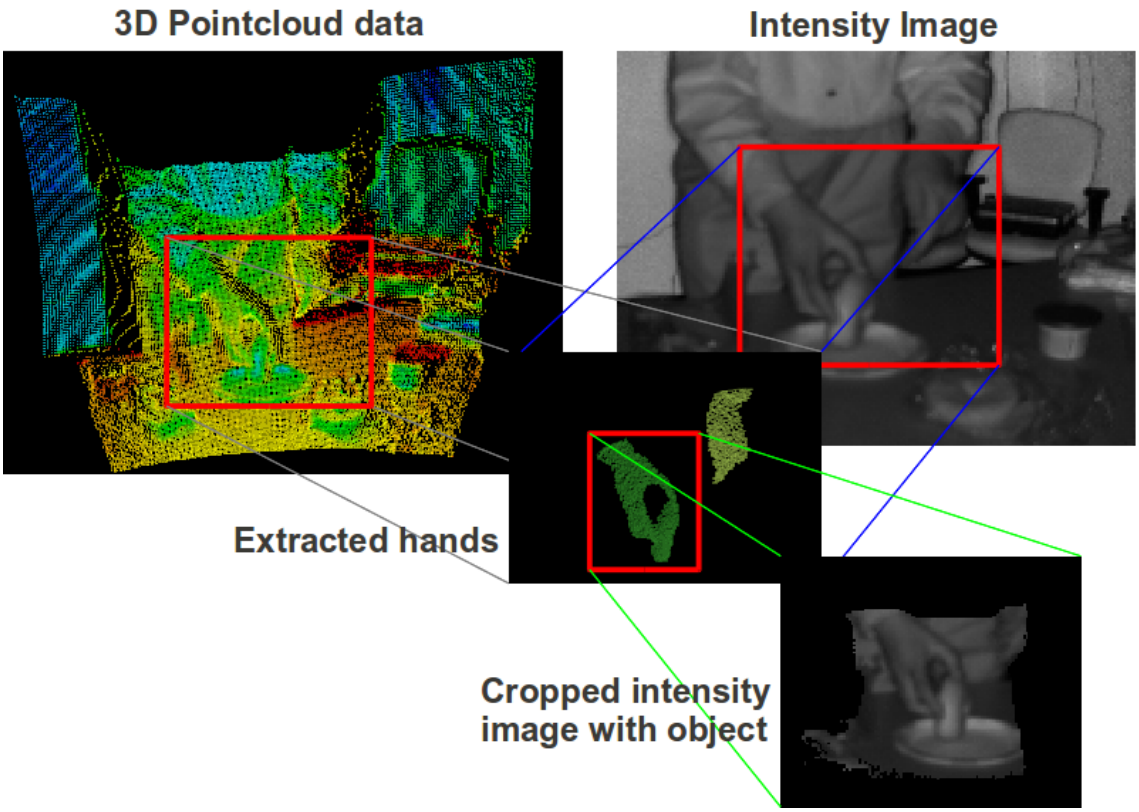


Figure 5.5: Merging occupied hand pointclouds with intensity image for object recognition.

for each object/tool class are trained over a separate training set of labeled data using a degree three polynomial SVM. We select the object labels from the classifier that gives the highest response for the case when $H_s = occ$. Due to the large amounts of occlusions when a merge event occurs, we select from the top $N = 4$ detection responses the most consistent object label from the previous frames (since it is unlikely that an object label changes when a merge occurs).

Visual filters can also be used to predict object labels, but one difficulty in using any scene classification technique for this kind of task is the small size of the objects of interest in proportion to the background. Such labels can be difficult to learn.

5.3.5 Building the Activity Tree

The previous steps tell us what objects or tools are grasped by each hand at each frame of the recording, and when objects are brought together and begin to be treated as a single object (a *merge* event). This provides enough information to build the activity tree, as shown in Fig. 5.6. The parser creates a new leaf node whenever one of the actor's hands (or tools held in the hands) come into contact with a new object. These nodes keep track of the time the object was first and last seen, and what the object was recognized as (using the HoG object recognition described in sec. 5.3.4.) If these objects are brought together, a new node is created with each of the original object nodes as children. This process gradually builds up tree structures.

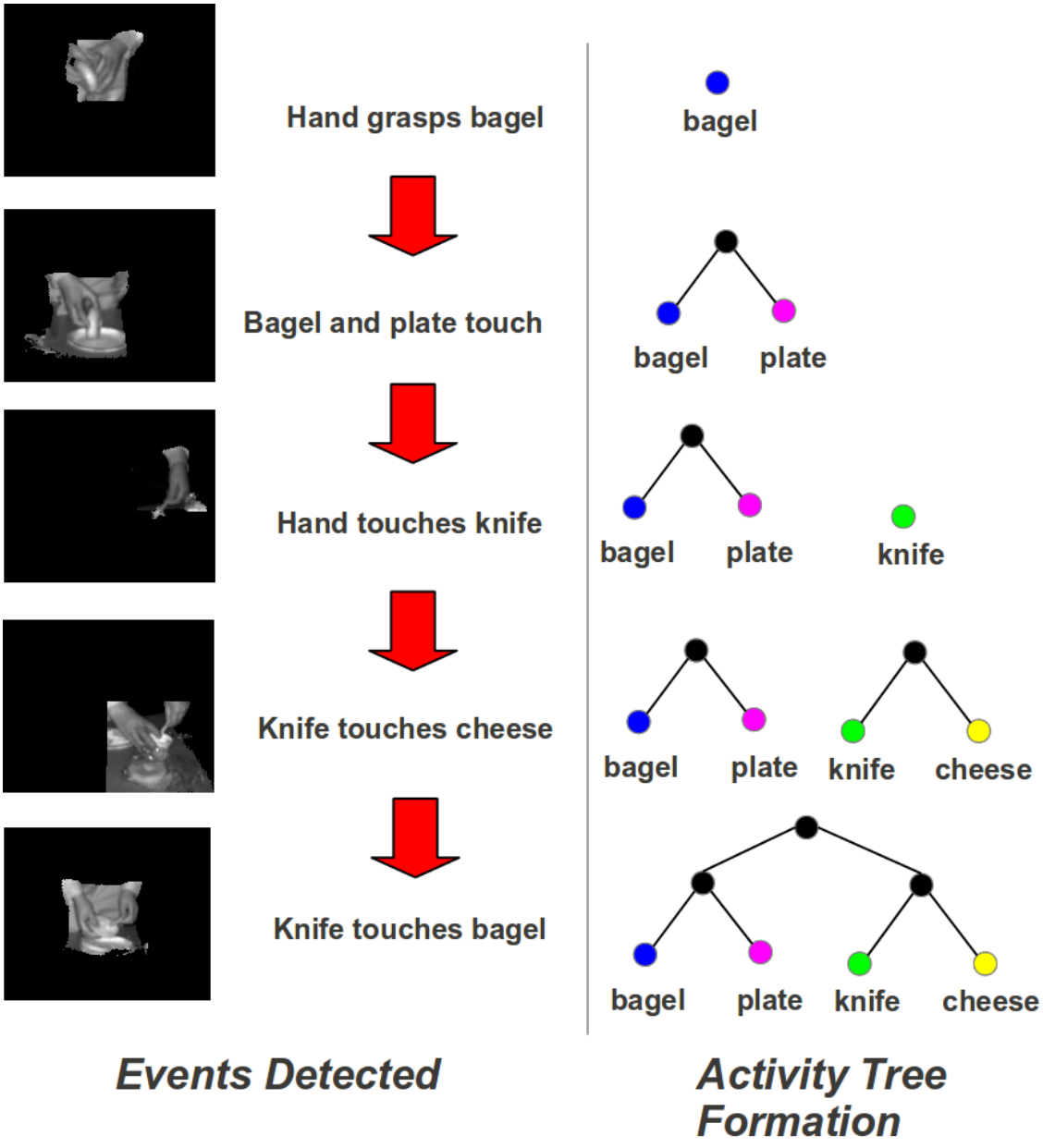


Figure 5.6: Creating an Activity Tree: (Left) Events and objects detected from SR4000 intensity images. (Right) Formation of an activity tree that parallels the events and objects occurrence, based on the defined action grammar.

Detecting when objects are brought together is not a foolproof method of recognizing when the objects begin to be treated as one combined object. One may, for instance, pick up two unrelated objects in one hand just to clear away a working space. In videos where this kind of event happens frequently, a better means of recognizing a significant, meaningful contact would be needed, or a better way of tracking where the objects ended up during a merge event.

To build a robust system, it would be useful to have several examples of each activity one wanted to recognize, and measure whether the activity tree from an unknown activity fell into this cluster.

There are many ways to perform any object manipulation activity. Certain aspects proved to be highly variable. Objects were frequently handled, set down, and picked up again, or passed from one hand to another, in an irregular way. A cutting action might be followed by a second cutting action if the results of the first cut were deemed unacceptable. The order of some of the actions differed from one trial to another.

Certain aspects of the activities however, were much more consistent. In order to correctly perform the tasks, certain objects needed to come together and not be separated again before the end of the activity. In the *Sewing a Toy* activity, for example, the pieces of felt needed to be joined together with the thread. Recognizing these critical merge events is crucial for understanding the activities.

5.4 Tree Edit Distance

An action grammar produces a grammatical tree structure representing the activity. Subtrees of this tree can be considered as subactions of the activity as a whole. The leaves of the tree are primitive actions. These activity trees will typically be noisy: missing subactions, including spurious ones, or mislabeling objects involved in the activity. Because of this noise and natural variability in the way activities are performed, comparing trees must be done in such a way that trees that are merely similar (not necessarily identical) can be matched. This is done with the Tree Edit Distance.

The Tree Edit Distance was introduced in [58]. The tree edit distance between two ordered, labeled trees is the minimal-cost sequence of edit operations that transforms one tree into the other. These operations are: 1. deleting a node and connecting its children to its parent, 2. inserting a node between a parent and a subsequence of its children, and 3. renaming the label of a node.

There have been several improvements in the time and space complexity of the algorithm in [59] [60] [61], so that the current best algorithm takes between $\Omega(m^2n)$ and $\Omega(m^2n^2)$ operations where m and n are the number of nodes in the trees being compared. For the small sizes of tree that are generated by action grammars, this comparison can be done very quickly.

Using the tree edit distance is critical for discriminating situations where the same objects are used in different ways. For example, if one were to put drafting instruments into a case to bring to school, the tree would consist of each instrument

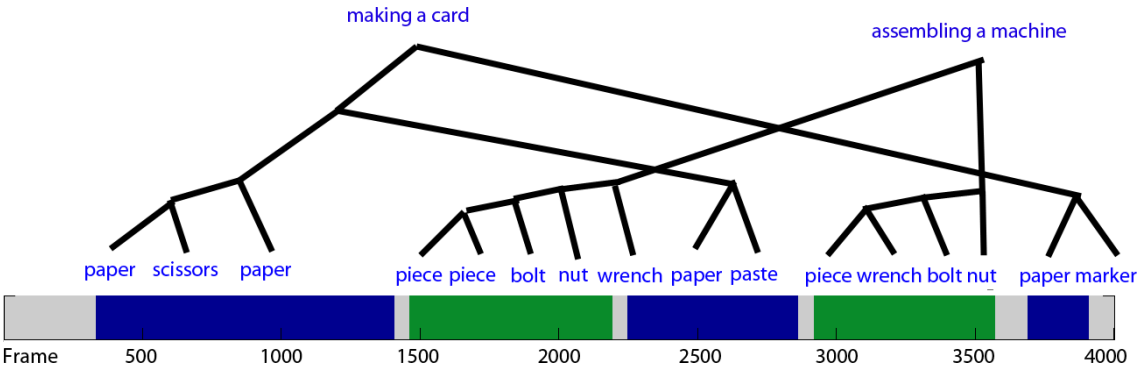


Figure 5.7: A complex interleaved sequence *Making Card + Assemble a Machine* can be cleanly separated into its component activity trees using the action grammar.

being merged with the case one by one. However, if one were to use the instruments for drafting, a more complex tree would be created, where the triangle is used with the pencil and paper, then moved out of the way, and so forth. Forming activity trees allows us to capture the structure of this interaction.

5.4.1 Separating Interleaved Activities

In many cases, an activity is an uninterrupted sequence of related events. In this case, segmenting the activity in a recording means simply finding the beginning and end point of the activity. However, there may be interruptions, in which the actor is trying to deal with more than one goal simultaneously. This results in actions that are mixed together. By parsing these actions, we are able to get a fine grained segmentation of the recording, identifying which actions belong to each activity, even when they are thoroughly mixed. To demonstrate this, several activities in the Complex Activity Dataset contain interleaved actions of *different* activities combined together. For example, the actor was asked to perform a cutting and pasting task (*Making Card*) and the *Assemble a Machine* task, interleaving the

actions for each activity. Because the activities involved separate objects, we were able to use the grammar to successfully separate out the actions for each task. This is shown in Fig. 5.7. As we will see in the experiments (sec. 5.5.3), the strength of this approach is highlighted when we are able to recognize such complex interleaved activities much better than a simpler approach when no action grammar is imposed.

5.5 Experiments

We report the results of two experiments that evaluate the performance of the action grammar in recognizing complex manipulation activities. We first derive theoretical bounds of the expected performance by inducing artificial noise in the terminals (sec. 5.5.2) and then evaluate the performance of recognizing activity trees over real data from the Kinect+SR4000 Complex Activity Dataset (sec. 5.5.3).

5.5.1 Experimental Procedure

For the experiment that explores the theoretical performance of the action grammar, we manually induced corruption in the input terminals of each activity tree from the Complex Activity Dataset in 2 ways: 1) by sampling from a uniform distribution of all possible object labels considered (except the ground truth) and 2) by consistently selecting the object labels from only one but a *different* activity tree for each associated object: e.g, if the activity was *Card Marking*, we will replace object labels consistently from another activity such as *Cooking Vegetables*. We considered corruption of the input ranging from 10% (almost correct) to 90% (almost

all wrong) and report the accuracy scores in interpreting the activity using the corrupted activity tree using the following procedure: for each level of corruption, we compute the edit distances for each tree, and take the ground truth identity of the smallest edit distance. We then count how many trees are correctly matched and report the accuracy score per level.

The next experiment evaluates the action grammar over 12 activities from the Complex Activity Dataset. In this part, we used a leave-one-out training procedure to train the object classifiers—for each test sequence, the remaining 11 sequences were used for training. Note that 4 sequences involving *Sewing* and *Assembling a Machine* are left out of the evaluation due to the fact that the object recognition simply failed as the objects of interests: pins, bolts, etc. are too small[5]. We then report the normalized tree edit distances of the resulting activity trees when they are compared with the ground truth, together with the amount of terminal corruption per sequence. As a comparison to highlight the contribution of the action grammar in building the activity tree, we also report the activity recognition performance when only the terminals are used to build a degenerate tree of depth 1 only (a flattened tree).

5.5.2 Results over Artificial Noisy Data

The accuracy scores over increasing degree of terminal corruption are summarized in Fig. 5.8.

[5]the specific sequences used and left out can be found at http://www.umiacs.umd.edu/research/POETICON/umd_complex_activities/

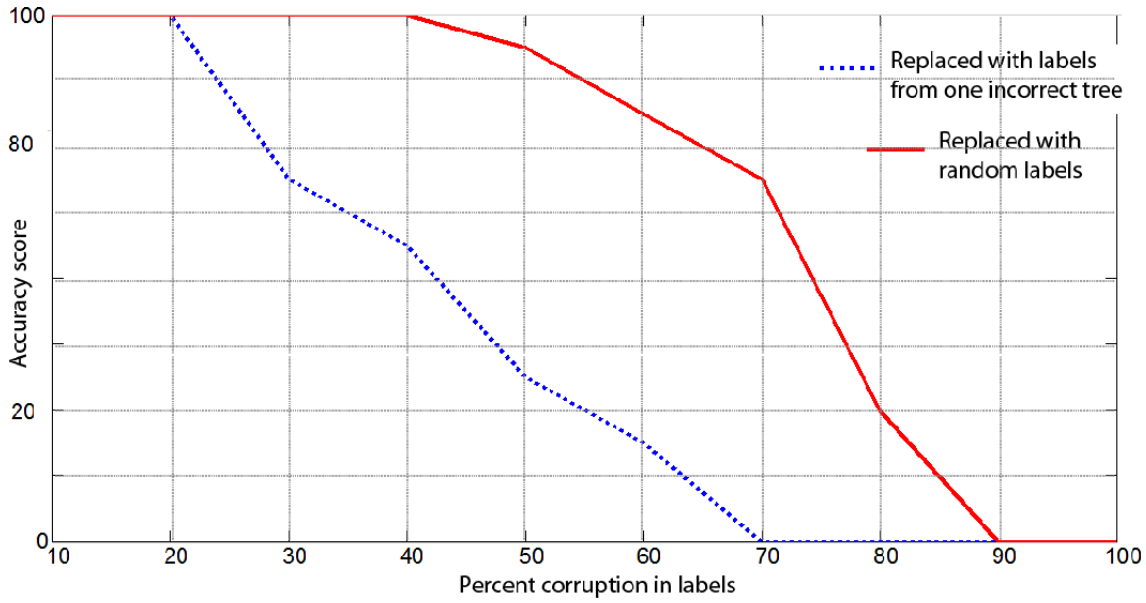


Figure 5.8: Accuracy scores with varying degrees of terminal corruption: 1) Randomly replaced object labels (red solid line) and 2) Replaced object labels consistently from another (incorrect) tree (blue dotted line).

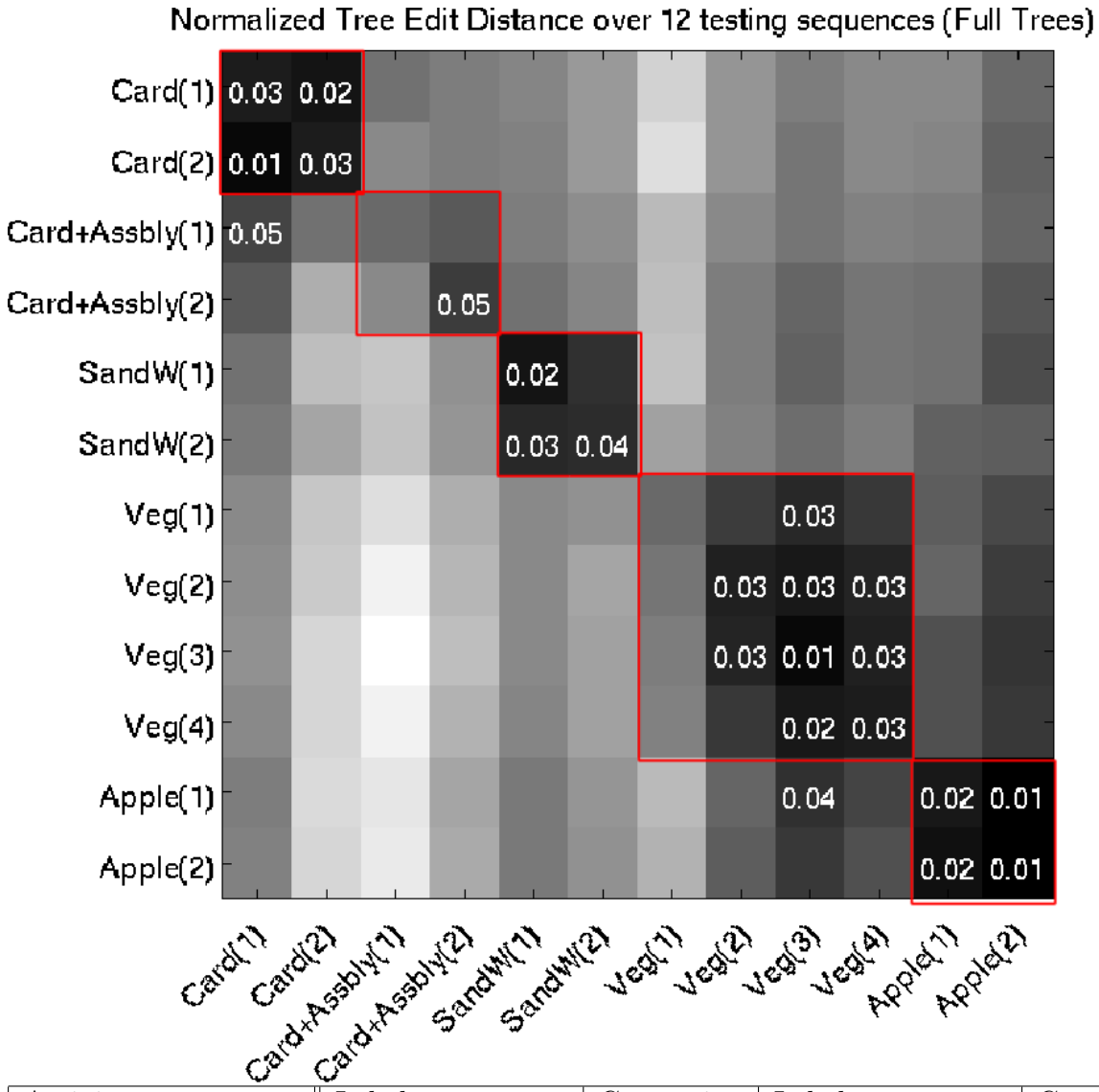
The activity trees are robust enough to handle the fact that no object detection method is completely accurate. In an attempt to characterize the behavior of tree matching in the presence of noise, we considered two possible causes of terminal corruption as described in the previous section. In the first case where the missed detections are completely random (the red solid line), the trees perform fairly well, accurately matching the true tree to the partially mislabeled tree in all cases until 40% of the labels have been replaced. In the second case (the blue dotted line), all the incorrect labels come from a single incorrect tree and so are consistent with each other. In this worst case scenario, the performance does worse, and errors in recognition show up when 20% of the labels are incorrect.

5.5.3 Results over Complex Activity Dataset

We summarize the matching performance for the 12 test activity trees in Fig. 5.9 and compare it against the baseline method of using terminals alone (Fig. 5.10).

In order to measure how well the activity trees could be used for activity recognition in real data, we computed the tree edit distance between each test tree and the ground truth for each of the activities. Each activity comes as a set containing at least 2 similar sequences. For example, *Card Making* has two sequences: *Card(1)* and *Card(2)*, performed by 2 different actors which introduces a small amount of variation within each activity set itself. In the confusion matrix above, the blocks of low edit distances along the diagonal for each activity set and higher distances elsewhere indicate that the activity trees are finding fairly good matches among the correct set of activities (Fig. 5.9 (above)). This performance is achieved in spite of the high levels of corruption in the terminals (Fig. 5.9 (below)) of between 36% to 62% that are sufficient to degrade performance (shown in the first experiments), which is indicative of the robustness of the approach in noisy real data.

By way of comparison, we flattened the trees so that all the nodes were at the same level (depth 1) and repeated the same experiment (Fig. 5.10). This effectively eliminates the effect of using the action grammar. In this case, the diagonal structure is much less evident, highlighting that the absence of the tree structure derived from the action grammar greatly reduces the ability of the system to find the right matches. This is especially true for activity sets that contains complex interleaved



Activity	Label	Corruption	Label	Corruption
Card Making	Card(1)	0.48	Card(2)	0.49
Card Making + Assemble Machine	Card + Assbly(1)	0.55	Card + Assbly(2)	0.48
Making Sandwich	SandW(1)	0.55	SandW(2)	0.36
Cooking Vegetables	Veg(1)	0.42	Veg(2)	0.62
	Veg(3)	0.54	Veg(4)	0.47
Cutting Apples[a]	Apple(1)	0.55	Apple(2)	0.42

[a] A subset of the *Cooking Vegetables* activities

Figure 5.9: (Above) Confusion matrix of normalized tree edit distances for each of the 12 test sequences. Lower values along the diagonals are better. Boxes indicate the diagonal blocks of interest for each set. (Below) Amount of corrupted terminals $[0, 1]$ per testing sequence. A value closer to 1 means more corruption.

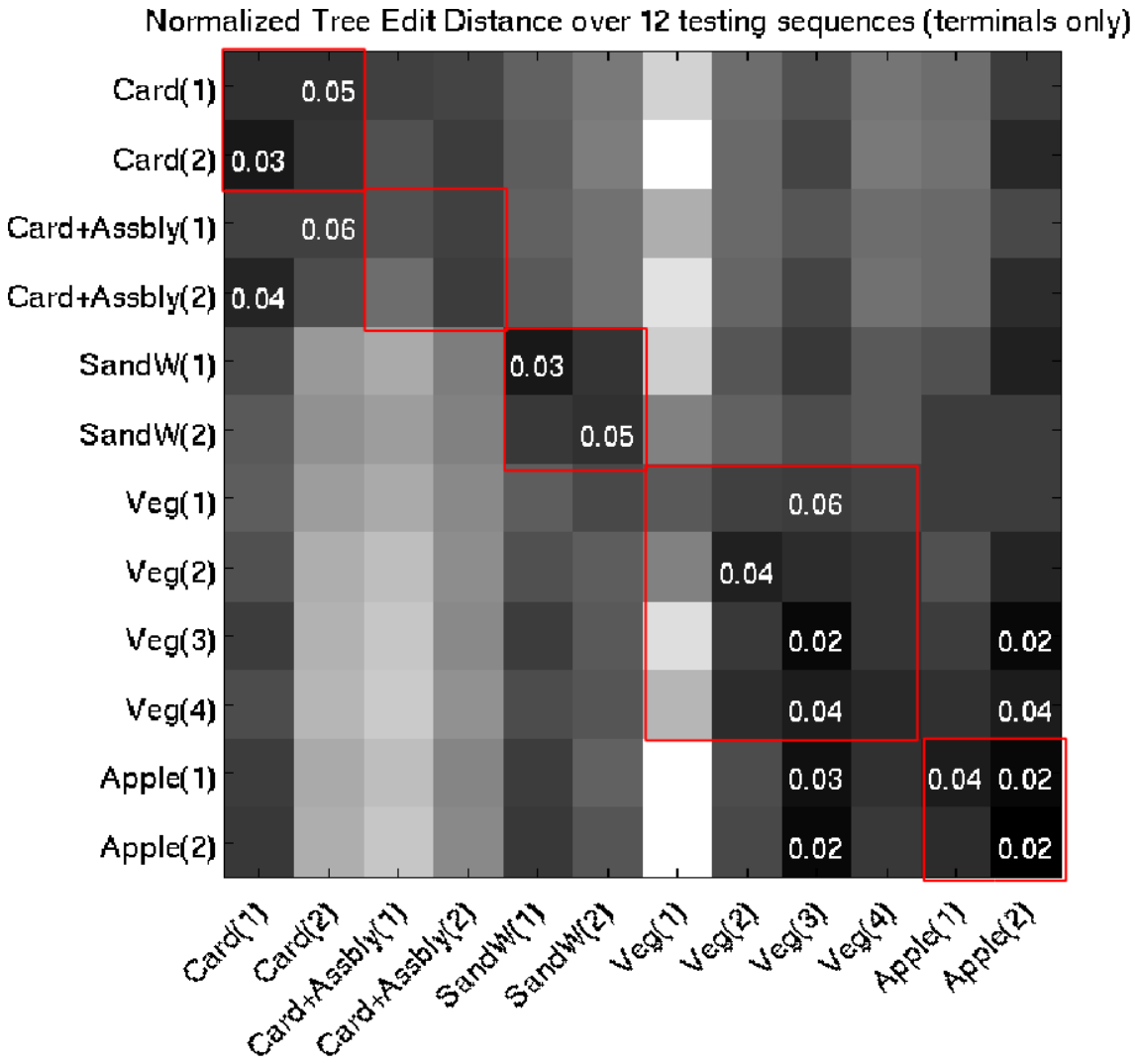


Figure 5.10: Confusion matrix of normalized tree edit distances when terminals are used alone. Lower values along the diagonals are better. Boxes indicate the diagonal blocks of interest for each set.

activities such as *Card Making + Assemble a Machine* and *Cooking Vegetables*. As was explained in sec. 5.4.1 and illustrated in Fig. 5.7, the ability of the action grammar in disambiguating complex interleaved activities is shown by the fact that the block diagonals for such activities display lowered performance when flattened trees are used (the tree edit distances are much higher within each block) compared to the ones when the full action grammar is used in the previous experiment (Fig. 5.9).

5.5.4 Results on Assembly/Disassembly Task

In order to test the more advanced action grammar, we created a new dataset consisting of the assembly and disassembly of a toy truck. We chose this subject because it included the use of a variety of tools, all the components were large enough to see (normal screws and nails are too small to be detected by the Kinect sensor), and the assembly could take place entirely on a table surface, in view of the sensor. Future depth sensors with higher resolution over a wider area will hopefully remove some of these restrictions for more practical tasks.

A rectified pointcloud image from this dataset can be seen in figure 5.11. The system was able to distinguish which steps were "assembly" type actions, where objects were merging together, and "disassembly" actions, where objects were splitting apart, with 82% accuracy. Most errors were due to treating a region of nearly touching screws as a single object, which then split into a pile and a single screw. It is not at all clear that this should even be treated as an error.

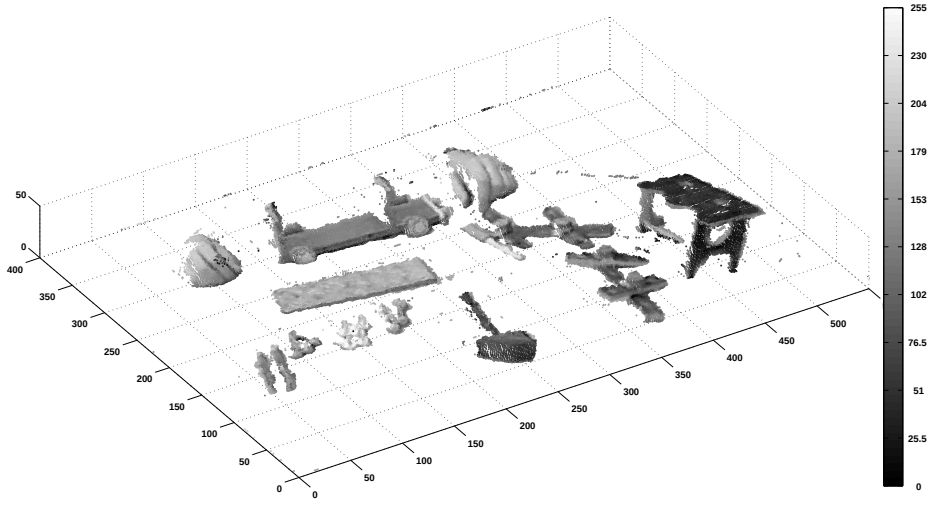


Figure 5.11: Pointcloud from Kinect sensor. This was converted to a 3-D cell occupancy map, cleaned up using 3D morphology techniques, and regions extracted. These regions were then tracked using a Kalman filter capable of handling splits and merges of tracks.

5.6 Conclusion and Future Work

Using a grammar of action to build activity trees appears to be a practical way to begin to parse complex activities. We are considering many possibilities for how to build on the current work. The grammar described here could easily be extended to include more specific patterns to be matched, creating a richer grammar that provides immediate information about actions and subactions. More traditional action recognition techniques could also be incorporated. For example, when a tool touches an object, we could determine whether the tool is actually performing its function on the object and transforming it, or just coming into contact with it. Object recognition could be improved by feedback from the tree structure, increasing the probability of detection for an object consistent with the rest of the tree. The

activity trees could also be used by a robot to emulate activities it has seen performed previously by humans, or even generated based on goals the robot is trying to attain.

Chapter 6: The Cyc Knowledge Base

6.1 History of the Cyc Project

Cyc (pronounced 'sike,' from 'Encyclopedia') is a project designed to provide common-sense knowledge to computer applications. It was begun in 1984 by AI researcher Douglas Lenat. His intention was to create an enormous knowledge base containing "the millions of everyday terms, concepts, facts, and rules of thumb that comprise human consensus reality." [103] Over 1000 person-years have gone into creating the software and entering the assertions that make up Cyc. [104] The current release includes 239,000 terms and over two million sentences relating these terms.

The original Cyc system is proprietary. In 2002, an open version of Cyc containing (at present) the inference engine, the concepts, and generalization links between them was released. However, without the assertions relating these concepts to one another in other ways, this skeletal version of Cyc is of limited usefulness. In 2006, ReasearchCyc, containing the full Cyc system (except for some of the natural language understanding components) was made available to universities wishing to explore the potential uses of the project. ResearchCyc is the version used for this work.

6.2 About Cyc

Cyc constants (including relations or predicates, attribute values, and instances and collections) are created in a representation language called CycL. Sentences in CycL relate these constants to one another, and can include any number of variables. (A query attempts to find bindings to the variables which make the sentence true.) CycL's grammar is Lisp-like, generally with a (*predicate subject object*) format, though there are also other arity predicates. Interaction with Cyc (for example commands like ASK and ASSERT) is handled through SubL, a subset of Lisp that is easy to implement in languages like C. There is also a Java API which maps closely onto parts of SubL, and a browser interface that allows easy access to these tools.

The Cyc inference engine handles *modus ponens* and *modus tollens* inferencing. It has some limited mathematical inferencing capability. Universal and existential quantification are handled through a strict separation between collections and instances, handled by the *genls* (generalization) and *isa* predicates. Special-purpose inference modules handle some of the most common situations, such as transitivity or reflexivity, in order to speed up performance. Various heuristics are used to guide the search to come up with answers to queries quickly.

Assertions in Cyc may be *monotonically* or *default true* (or *monotonically* or *default false*). *Monotonically true* statements that contradict other *monotonically true* statements generate an error when they are asserted, preventing the assertion. *Default true* statements, on the other hand, may contradict one another, and

various heuristics are used to determine how to resolve the truth value. Every assertion records its *support*, either a particular user who entered the assertion or the automatic extraction (for example, from Wikipedia) that created it.

Context is handled mainly through the use of *microtheories*. Each assertion is contained within one or more microtheories. By limiting a query to a particular microtheory, assertions in other microtheories are ignored. For example, each time my system observes a new image, facts about that image are asserted in their own microtheory. This prevents facts from one image from polluting the reasoning about another image. Microtheories are arranged in a directed acyclic graph, so that one microtheory can include a set of microtheories that fall under it.

The Cyc system has failed to find wide acceptance to date for a variety of reasons. Some of the design decisions, such as building the system in a variant of Lisp, awkward handling of context, and little support for probabilistic reasoning, are quite different from how a similar system might be built if it were started today (compare to ConceptNet). For years the memory, processing speed, and disk space of personal computers was too small to be able to make use of Cyc, though this is no longer really much of an issue for many applications on 64-bit machines. (Most queries that I have tried come up with a list of answers in less than one minute.) The idea of deliberately crafting such a large edifice by hand by trained employees is very different from the automatic learning approaches currently in favor. The coverage of facts is uneven and eclectic. While a complete system would be very useful, a system with gaps is simply frustrating. Because there are multiple ways to encode knowledge, finding the right way to ask the question requires some explo-

ration. Because any rule can potentially interact with any other rule, adding to the knowledge base safely can be tricky.

6.3 How Cyc is used in this project

When the system is supplied with a video, it runs scene parsing, material parsing, and activity recognition on the frames of the video. The results of the scene and material parsing are divided into contiguous regions using image morphology techniques. This generates a list of objects in the scene (each region is assumed to contain one or more objects from the detected class) a list of materials, and a list of detected activities (along with their subtrees). These descriptive labels for the scene allow the software to make certain assertions in a Cyc microtheory created especially for the dataset currently being analyzed. It can assert that an object from each of the detected classes is present in the scene, and some rudimentary spatial relationships between objects, such as that one object is above another or that from the perspective of the camera an object is to the left or right of another object. Assertions are made regarding the material of parts of objects, and about the presence of a camera at the scene to capture the image and the existence and properties of the digital image.

Additional assertions are made relating how classes tend to interact in image space. For example, if a physical object of some kind is mostly above and adjacent to a horizontal planar object (the ground, a floor, any type of terrain), we can assume that the object is resting on the surface (unless the object is currently airborne).

When a class is defined as 'automobile,' for example, Cyc knows that an automobile is a physical object, and so is able to infer, using this rule, that the automobile is resting on the object classified as 'road'. Assertions are made about the relative position of the sky, a class that is nearly universally present in outdoor scenes.

These assertions exist within the context of hundreds of thousands of related assertions. The ability to make inferences using this information allows the system to answer a wide variety of queries about the image. For example, for many classes, the physical parts of the object are defined. So queries relating to these parts can be answered, even though these parts have no detectors defined for them. The user can also ask for the names of potential objects whose parts are in the scene, or for all objects typically made of some material in the scene.

Cyc has large gaps in its knowledge, but from our understanding of meta-information about the dataset and the types of scenes it represents, we can define assertions filling in some of these gaps at the time the labelled training images are created. Because these statements are defined in terms of the world as a whole, they extend the abilities of Cyc and improve results on future queries.

6.3.1 Improving Material Labeling Using Object Labeling (and vice versa)

When the object and material detectors are run on evaluation images, statistics are collected on how often each pair of object/material labelings occurred and what the true label was in each instance. Using these statistics, given an object labeling

and a material labeling for a scene, the system builds a map that has defined for each region the probability of each object/material pairing. However, certain object material pairings may never come up in the training samples. In this case, we rely on prior knowledge about what materials objects are made of stored within Cyc to fill in those probabilities. If we assume the information in Cyc is 100% true, this drives the probability in odd cases such as the same region being labelled 'metal' and 'tree' to the two possibilities that either the object is made of metal, or it is a tree, but not both.

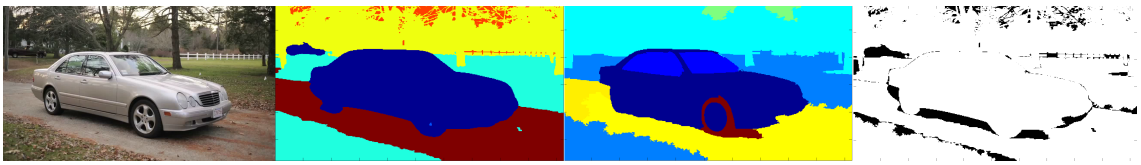


Figure 6.1: [left]original image. [center left] object class labeling(lawn, road, trees, car, sky). [center right] material labeling (rubber, metal, grass, gravel, foliage, sky, glass). The regions labeled in black are incompatible between the material and object labelings. At least one of the labelings must be incorrect in these areas. Note that this cannot catch errors such as the rear tire being labeled as metal rather than rubber, because some parts of a car are metal.

6.3.2 Ambiguous Labels

The material and object parsing step returns probability maps for each class for the image. Most of the time, these maps are close to 100% or 0%. When two classes both have substantial probability (both are between 25% and 75%) the system cannot confidently assign any of the labels. In this case, one possibility would be to assign no label to this region, reflecting our uncertainty. Another possibility is to assign a label which would be correct if *either* label happens to be correct. This

can be done by finding a class which includes both of those labels as subclasses. Of course, assigning the label 'thing' to everything in the scene would be technically correct[1] but unhelpful. Instead, the system assigns the nearest superclass—a class which is a superclass of both labels but has no subclasses which fit that definition. For example, a slightly unfocused green region which is ambiguously labeled 'foliage' and 'grass' would be re-labelled 'plant matter.' This allows us to conclude certain facts about the region (that it contains living things, that it is not manmade, etc...) despite our uncertainty about the label.

6.3.3 Answering Queries

Based on the information in these assertions and the rest of the Cyc database, the system can answer queries regarding the scene. These queries must be asked in CycL, a formal Lisp-like language.[2] However, the results can be automatically translated into (somewhat stilted) English using templates that are defined for most Cyc terms. In addition, the steps of justification are also given in English. In Chapter 7, English queries, their CycL equivalent, the automatic translation of the CycL query, the answers, and some justification are listed for example queries.

6.3.4 Image Translation

Using the knowledge about the scene stored as Cyc assertions, we can try to reconstruct a simplified representation of the original image. This is discussed in

[1] The best kind of correct.

[2] Internal research at Cycorp has been done on automatically translating between English and CycL, but this is not yet possible for general queries.

chapter 8.

6.4 Conclusion and Future Work

One interesting area not explored in this work is supplementing the world knowledge in Cyc with information observed about the world (rather than just adding knowledge about a particular scene.) If we trust our detector more than we trust the information currently in Cyc, this could be used to modify or supplement the Cyc statements in the overall model of the world, which is reused in analyzing future scenes. (Currently, all assertions made about a scene are made in a microtheory which is unable to effect anything outside the microtheory.) Tentative statements about the relationship of objects and materials not yet in Cyc could also be derived from the label or even the test images, and be used to help in future problems. Modifying general statements about the world in Cyc based on the limited parts of the world seen in particular datasets is problematic, however—it may be that all of our data comes from a the United States, and doesn't apply to other countries, or any number of other differences in context. Finding ways to do this safely (i.e. not introducing false statements which could pollute the truth of any future conclusion) and effectively is an interesting area to explore. Ultimately, one would like to be able to use machine learning techniques to build up a world model similar to that contained in Cyc from scratch.

	lawn	road	tree	car	sky
rubber	.024	**.720**	.024	.016	.016
metal		.018			
grass		.027			
gravel	.002	.072	.002	.001	.001
foliage		.027			
sky		.018			
glass		.009			

	lawn	road	tree	car	sky
rubber				X	
metal				X	
grass	X				
gravel		X			
foliage			X		
sky					X
glass				X	

	lawn	road	tree	car	sky
rubber				.176	
metal				.004	
grass	.010				
gravel		**.791**			
foliage			.010		
sky					.002
glass				.002	

Table 6.1: Using information from Cyc about what materials objects are made of, the number of possible pairings of object and material are limited. Of the possible pairings, we can act as if the material and object probabilities are independent and choose the label with the highest calculated probability. The calculation shown is for a pixel in the shadow of the front tire in the image above. Originally [top], it is misclassified as 'rubber road.' When impossible pairings are eliminated using the information from Cyc [center] the correct label [bottom] 'gravel road' is found. (Values smaller than .001 are omitted for clarity.)

Chapter 7: Answering Queries

The following are examples of CycL assertions automatically made by the system when presented with images, and the replies to selected queries about the observed scene. The overall process of labeling videos with visual filters, material segmentation, and action grammars and making assertions in Cyc based on these observations is explained in the previous chapter.

The system as a whole was applied to a short instructional video (found online) of how to replace a tire. The video was chosen because it was in high definition, from a single camera, and showed an activity that included both assembly and disassembly subtasks. Training labels were made on five frames of the video and applied to each frame of the video. Because the camera never changes direction and the scene is a simple one, this allowed very high pixel labeling accuracy over the full video. Lacking depth data, we had no way of reliably detecting object "merge" and "split" events, so this information was supplied by hand.

Cyc already contained a detailed breakdown of the steps of changing a tire. By matching objects involved in the steps of the script to subtrees of the activity tree, the system was able to associate the two representations and recognize that the video as a whole contained a tire-changing activity.[1]

[1]It should be noted, however, that this coincidentally happened to be a well-developed area

This only touches on the possibilities of such a system. I carefully refrained from adding new information about the world specific to the dataset on which I performed the queries, only making assertions that can be made for any image. I did this to keep from answering the very questions I wanted to ask, which would make the test less meaningful. Realistically, however, at the same time one created training labels for a particular dataset, one could also assert a great deal of information about that dataset, putting it into context. These assertions would also be able to interact with the specific assertions about detected classes, enabling response to a wider range of queries. Cyc at present has little notion of relative positions. It is impossible to assert in Cyc that "the man is in front of the car from the viewpoint of the camera." One future task I would like to work on is to create a program to take a map and automatically generate Cyc assertions that capture this information.

X and Y stand for each pair of detected objects in the scene. They have been arranged as tables presented horizontally to make them easier to read and interpret. When Cyc performs an inference, it can tell the steps of reasoning by which it came to a conclusion. These statements in CycL are automatically translated into English. The assertions and queries, also written in CycL, are also automatically translated into English.

The following was the description of events in changing a tire I was attempting to match. Detecting action motions such as wrenching, pushing and pulling were not attempted.

---

of Cyc's knowledge, and that few activities are represented within Cyc to this level of detail at present.

- A. hand touches wrench
- B. wrench touches (lugnut + wheel + car)
- C. (wrenching action occurs)
- D. wrench is removed from (lugnut + wheel + car)
- E. hand separates from wrench
- F. hand touches (lugnut + wheel + car)
- G. lugnut separates from (wheel + car)
- H. hand separates from lugnut
- I. hand touches (wheel + car)
- J. (pulling action occurs)
- K. wheel is removed from car
- L. hand separates from wheel
- M. hand touches sparewheel
- N. (pushing action occurs)
- O. sparewheel is attached to car
- P. hand separates from sparewheel
- Q. hand touches lugnut
- R. (screwing action occurs)
- S. lugnut is attached to (sparewheel + car)
- T. hand separates from lugnut
- U. hand touches wrench
- V. wrench touches (lugnut + sparewheel + car)
- W. (wrenching action occurs)
- X. wrench separates from (lugnut + sparewheel + car)
- Y. wrench separates from hand

  The rules refer to the following description of the action grammar:

  1. When an object is touched (either directly by a hand or by a tool being used by the hand), a new node is created.

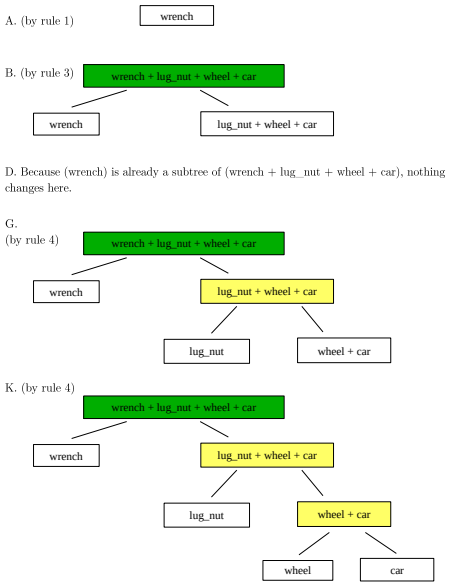
  2. When an object transforms from A to B, a new node labeled B is created and attached as a subtree to node A.

  3. When multiple objects (A,B) combine into one object A+B, a new node labeled A+B is created the subtrees associated with A and B are attached (as subtrees) to A+B.

  4. When one object A+B separates into multiple objects (A,B), new nodes A and B are created and attached as subtrees to the node A+B.

  5. In addition to the above, a new node can be created for an object already used in the activity, if it is being used as part of a new assembly or disassembly.

Figure 7.1: A frame from the ‘changing a car tire’ video.



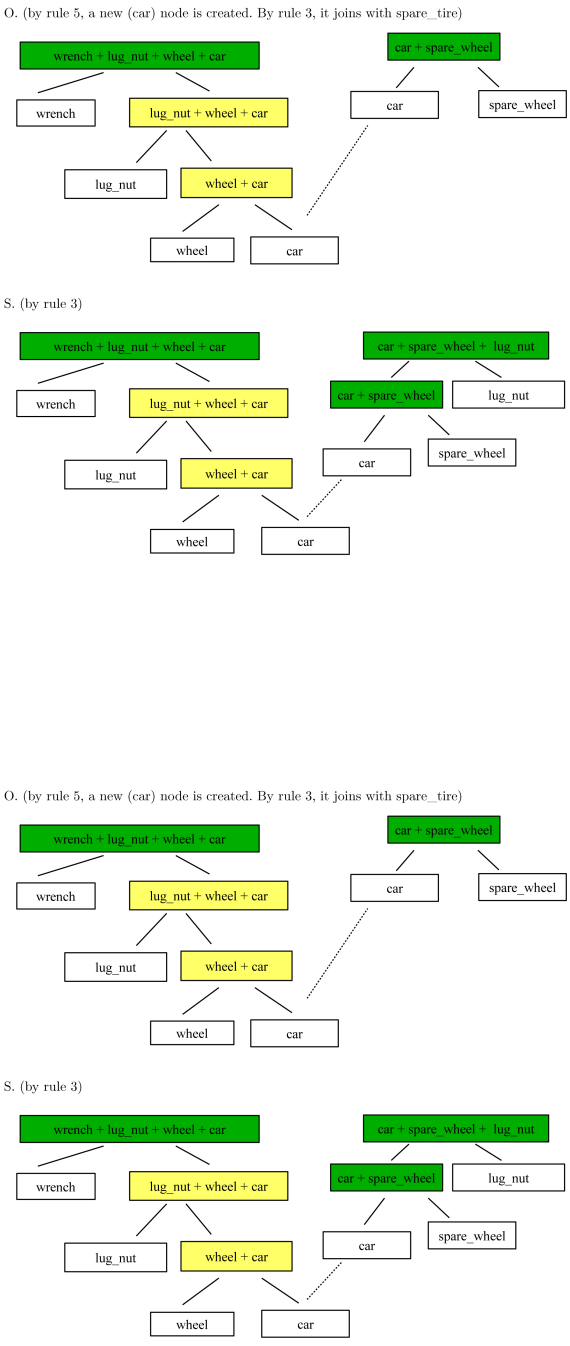


Figure 7.2: The activity tree for changing a car tire. Assembly nodes are colored green, and disassembly nodes are colored yellow. Recognizable actions (such as the wrenching action in C) can be used as further labels to make tree recognition more robust. Nodes involving the hands have been left out of the diagrams for clarity.

Intended English meaning	CycL	Auto. translation from CycL
There is a scene called Scene001.	(isa Scene001 Environment-Generic)	Scene001 is an environment.
X is in scene001.	(objectFoundInLocation X Scene001)	X is located in Scene001.
X is near Y.	(near X Y)	X is near Y.
X is next to Y.	(adjacentTo X Y)	X is adjacent to Y.
If X is a solid physical object, then unless the object is floating in the air, the following can be determined by the relative locations of the bases of objects. The assumption is made that the camera is level with the ground and not resting on the ground.		
If X is a solid physical object, X is (above or behind) Y.	(implies (isa X Landmark-Underspecified) (or (above-Generally X Y) (behind-Generally X Y))))	If X is landmark underspecified, then it is either above Y or it is behind Y.
If X is a solid physical object, X is (below or in front) of Y.	(implies (isa X Landmark-Underspecified) (or (below-Generally X Y) (inFrontOf-Generally X Y))))	If X is landmark underspecified, then it is either below Y or it is in front of Y.
There is a camera called Camera001.	(isa Camera001 DigitalCamera)	Camera001 is a digital camera.
Camera001 is at the scene.	(objectFoundInLocation Camera001 Scene001)	Camera001 is located in Scene001.
X is in front of Camera001.	(inFrontOf-Generally X Camera001)	X is in front of Camera001.
There is a digital image file called Photo001.	(isa Photo001 DigitalPhotograph-File)	Photo001 is a digital photograph.
Photo001 is a recorded digital image.	(isa Photo001 RecordedVisualImage)	Photo001 is a recorded digital image.
The scene is in Photo001.	(visuallyDepicts Photo001 Scene001)	Photo001 visually depicts Scene001.
Color001 is a color.	(isa Color001 Color)	Color001 is a color.
Color001 has red value Z1.	(colorHasRGBRedValue Color001 Z1)	colorHasRGBRedValue holds of Color001 and Z.

continued on next page

Continued from previous page		
Intended English meaning	CycL	Auto. translation from CycL
Color001 has green value Z2.	(colorHasRGBGreenValue Color001 Z2)	colorHasRGBGreenValue holds of Color001 and Z.
Color001 has blue value Z3.	(colorHasRGBBlueValue Color001 Z3)	colorHasRGBBlueValue holds of Color001 and Z.
A color of X appears to be Color001.	(significantColorOfObject X Color001)	X has Color001 as a significant color.
X has a physical part made of material W.		

What tools were used in the movie and what were they used for?	
Question in CycL	**automatically translated from CycL**
(sitTypeRequiresRolePlayerOfTypeInSubSitType ChangingAnAutomobileTireScript ?Y ?X ?Z)	What values of Y, X, and Z are there such that every changing an automobile tire script has an instance of Z as a sub-event in which some Y plays the role X?
Answers in CycL	**automatically translated from CycL**
deviceUsed TireIron (RemovalFn LugNut)	Every changing an automobile tire script has a lug nut removal as a sub-event in which some tire iron plays the role device used.
supportingObject (SupportingFn Automobile) JackLiftingDevice	Every changing an automobile tire script has supporting an automobile as a sub-event in which some jack plays the role device used.
providerOfMotiveForce (SubcollectionOfWithRelationToTypeFn LoweringAnObject objectLowered Automobile) JackLiftingDevice	Every changing an automobile tire script has lowering an automobile as a sub-event in which some jack plays the role device used.
providerOfMotiveForce (SubcollectionOfWithRelationToTypeFn LiftingAnObject objectRaised Automobile) JackLiftingDevice	Every changing an automobile tire script has raising an automobile as a sub-event in which some jack plays the role device used.

What are the steps involved in changing a tire?	
Question in CycL	**automatically translated from CycL**
(characteristicProperSubEventTypes ChangingAnAutomobile-TireScript ?X)	What values of X are there such that all changing an automobile tire script has X as a characteristic sub-event?
Answers in CycL	**automatically translated from CycL**
(TotalityOfSubEventsOfTypeFn ChangingAnAutomobileTire-Script(RemovalFn LugNut))	removing all of the lug nuts from the tire
ChangingAnAutomobileTire	changing an automobile tire
(SubcollectionOfWithRelationToTypeFn LoweringAnObject objectLowered Automobile)	lowering an automobile
(SubcollectionOfWithRelationToTypeFn LiftingAnObject objectRaised Automobile)	raising an automobile

What other parts of the car might need to be repaired?	
Question in CycL	**automatically translated from CycL**
(and (properPhysicalPartTypes Automobile ?PART)(arg1Genl RepairingFn ?X)(genls ?PART ?X))	What values of X and PART are there such that the first argument to RepairingFn must be a subtype of X, PART is a subtype of X, and every car has some instance of PART as a proper physical part?
Answers in CycL	**automatically translated from CycL**
AutomobileHeadLight (51 other actual automobile parts, and 9 stranger answers like TopOfCar, LeftSideOfCar and InteriorOfCar)	automobile head light
Sample Justification	
(genls Automobile WheeledTransportationDevice) (properPhysicalPartTypes WheelAndAxle Axle) (properPhysicalPartTypes WheeledTransportationDevice WheelAndAxle)	A car is a type of wheeled vehicle. Every wheel and axle has some axle as a proper physical part. Every wheeled vehicle has some wheel and axle as a proper physical part.

What is an automobile used for?	
Question in CycL	**automatically translated from CycL**
(artifactTypeCommonlyUsedWithSitType Automobile ?X)	What values of X are there such that cars often play a role in X?
Answers in CycL	**automatically translated from CycL**
(TransportViaFn Automobile)	transportation by means of cars
CarBombing	car bombing
(SubcollectionOfWithRelationToFn Stealing deviceTypeUsed Automobile	stealing with a car

Why is the car on the road, rather than the grass?	
Question in CycL	**automatically translated from CycL**
(conceptuallyRelated Automobile Roadway)	Is it true that the concept "car" is conceptually related to the concept "roadway"?
Answers in CycL	**automatically translated from CycL**
True	Yes
Sample Justification	
(conceptuallyRelated Automobile Roadway)	The concept "car" is conceptually related to the concept "roadway".
(prototypicalPathMediumTypeForTransportationDeviceType Roadway RoadVehicle)	The typical medium for movement by motor vehicles is via roadways.
(genlInverse prototypicalPathMediumTypeForTransportation-DeviceType conceptuallyRelated)	If the typical medium for movement by some type of transportation device is via some path type, then the concept "that type of transportation device" is conceptually related to the concept "that path type".
(genls Automobile RoadVehicle)	A car is a type of motor vehicle.

Chapter 8: Translation of Images from One Style to Another

8.1 Introduction

One of the tasks Bloom's Taxonomy lists under comprehension is translation. This is illustrated by particular educational objectives, including "translation from one level of abstraction to another" and "translation from one symbolic form to another." [11] For writing, translation means representing the text in another language. It is not clear, however, exactly what translation should mean when referring to visual media. Is Baz Luhrmann's film *Romeo + Juliet*, set in California in the 1990s, a "translation" of the original play into the new setting? Bloom gives example of translating from a written description to an image or vice versa, but does not explicitly provide any examples of translation from one visual medium to another. One possible answer is that a translation is a rendering of the scene in a new artistic style. There are many approaches in the computer graphics literature for 'non-photorealistic rendering' (NPR) which take a photographic digital image and transform it into a new painterly style, such as pen and ink, watercolor, or even stained glass. (For a good, if slightly dated, survey of these techniques, see [99].)

All of these techniques, however, leave the proportions and projection of the representation essentially unchanged. Evidence from studies of children's drawings and nave adults (untrained in making art) shows that the human process of making images works very differently than any of these techniques. As Martin Gardner writes, "[A] child wants...and is perhaps driven to invent, graphic equivalents for

those categories that occupy her thought processes; and so it becomes natural for her to develop a formula, or prototypical schema, which can represent or stand for the full range of instances of this category." [100] It is only with artistic training—education about the effects of perspective, tricks for seeing the scene as blobs of color rather than individual objects, measuring and copying proportions—that artists are able to reproduce images in true proportion, the way that is most easily realized by automatic computer techniques.

Figure 8.1 shows 21 images of a particular street scene drawn by children ranging from age 3 to age 13. The oldest children (one of the eight-year-olds, one of the 10-year-olds, and both children 11 and older) attempted to capture the strong perspective on the street by drawing converging rather than parallel edges. The younger children's drawings all draw the street as seen from directly above. Yet none of them have ever seen this particular street from that perspective. Instead, it seems like the drawing process the children are engaged in is something like the following:

1. Look at the picture.
2. Notice and recognize the most salient objects. (This may be very different for different observers. Johnny (age 4) apparently noticed the vehicle, the fence, and the sidewalk, while Maggie (also age 4) noticed the two figures, the tree, two buildings, and the sidewalk and road.)
3. Use pre-learned techniques for representing these objects, introducing only small variations to match what is present in the scene. (For example, Elena (age 6) draws conventional stick figures, but extends the arm of one around the shoulder of the other to match what she believed she saw in the image.) The younger children have learned such a technique for drawing a house but not for a building, but use this representation because it is conceptually the closest they have. All but one of the children age 5 or older also copied the

road markings, perhaps because they had the ability to represent them more or less accurately.

Even in the drawings of the older children, traces of this method are still present. In the 11 and 13-year-old drawings, attempts are made to capture the perspective on the buildings. Yet certain angles that ought to be acute in the perspective projection are instead drawn as right angles. It seems likely that this is due to the fact that the artists knew these angles must be 90 degree angles because of experience with buildings in the past, and that this knowledge informed their drawing, forcing accuracy in other areas to be compromised as the artist tried to reconcile the two implicit perspectives. Daniel (age 11) spontaneously commented as he was doing the drawing that he "didn't know how" to draw the cars from an angle.

In 1997, J. Willats performed similar experiments in which young children depicted a scene of a die. They often did this by including all sides of the die they could see within a single containing rectangle. He wrote, "it suggests that analogies between picture production by either photography or computer graphics and human picture production... are grossly oversimplified.... pictures can be derived from either object-centered descriptions or viewer-centered descriptions... the presence of characteristic anomalies...may provide the only available evidence about the nature of the production process." [101]

Discussing these experiments, Fredo Durand wrote, "This might seem like a very odd example due to the lack of skill. In fact, this is a caricatural but paradigmatic demonstration of a very fundamental principle of depiction: Depiction is not about projecting a scene onto a picture, it is about mapping properties in the scene to properties in the picture. Projection happens to be a very powerful means to obtain relevant mappings, but it is not the only one, and it is not necessarily the best one." [102]

8.2 Automatic Generation of Child-like Visual Representation

A simplified model of the production processes that children employ in making drawings of a scene was easily implemented within the automated image comprehension framework. Using the scene parsing software, a description of the objects in the scene and some information about their spatial relationships is stored in a microtheory in the Cyc ontology. A pre-stored schema for generating images in each of those categories is associated with it. (In this case, it is simply a stored bitmap, but a more realistic model would store it as a series of hand motions to create particular shapes with strokes of the pen and the spatial relationships of those shapes.) Using these schemas and the information about how the objects are distributed around the image relative to each other, a child-like representation is then generated. An example generated image and its associated input is shown in Figure 8.2.

Although it doesn't model everything a child is doing as she creates a depiction of a scene, it does capture one important aspect of that process. The point here is not to create a new way of generating cute images, but to suggest a new approach to NPR rendering that makes use of this important feature of human art. Consider the following goals of various artists:

- A cartoonist may want to reduce the number of individuals in a crowd to simplify the depiction.
- A painter may concentrate fine brushstrokes on the face of the focal character, while leaving the background depicted with broad strokes.
- A painter who wants his charming landscapes to be hung above the couches in people's homes may leave out garbage blowing down the street and modern automobiles present in the scene.
- A fantasy artist may wish to depict horses with wings.

Any NPR system that only considers the input image in terms of simple primitives

such as contours and regions of color would be incapable of being directly extended to handle these kinds of problems. A system including image comprehension with a semantic knowledge base, however, could be extended to deal with any of these problems in a fairly straightforward way.

8.3 Conclusion and Future Work

While this system has some interesting and unique capabilities, there is still an enormous amount of work to be done before image parsing and the knowledge base are usable in more than a few hand-tailored situations. Scene and material parsing have low accuracy rates on general images; they only work well when trained on one part of a limited dataset and tested on another part of the same dataset. The action grammar system relies on being able to accurately track the 3D position of every object in the scene, being trained to recognize the particular objects involved in the activity, and having seen other examples of the activity before. Cyc is missing huge areas of knowledge about the world, is incapable of most spatial or probabilistic reasoning, and requires learning a new language in order to query it. The image generation technique is little more than a parody of an artist's method.

Nevertheless, I feel this represents an architecture on which a truly useful system can be built. The parts complement each other, and enable new ways of thinking about what problems are even in principle solvable with existing technology. With work on improving the scope and robustness of the components, it should be possible to build a system along these lines that can continuously observe the world and use those observations to build its semantic model more and more completely over a long period of time. We will be able to interact with such a system in ways more like we would another person rather than a tool, expecting it to be able to act in ways consistent with an understanding of what is being asked of it. As new methods are developed for learning the 3D structure of the world, recognizing the

function of unknown objects, for guessing the motivations of actors in the scene, for understanding language, all of these can be added as extensions of an already existing architecture, feeding their insights into a common pool of knowledge where they can be combined and reused for many other purposes.

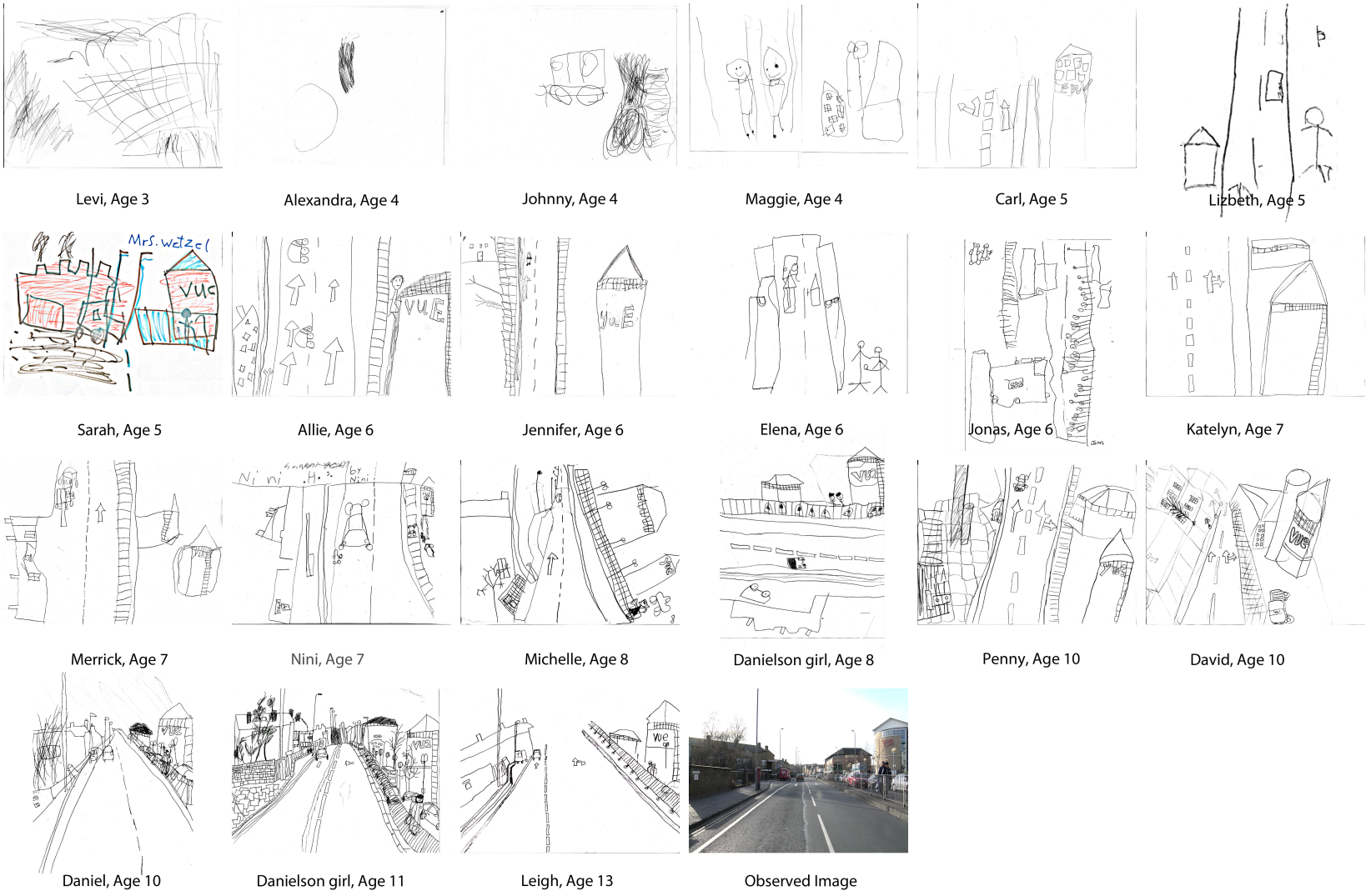


Figure 8.1: Drawings by 21 children of a street scene. Childrens' names and ages are shown. Some names have been changed to protect privacy.

Figure 8.2: [right] Image generated from pre-drawn templates and certain facts about what objects are present in the image and their spatial relationships stored as assertions in Cyc. [left] Observed image. [center] Parsed image.

Bibliography

[1] http://machinamenta.blogspot.com/2012/08/pareidoloop.html

[2] Searle, John R. "Minds, brains, and programs." Behavioral and brain sciences 3.3 (1980): 417-457.

[3] Jackson, Frank. "Epiphenomenal qualia." The Philosophical Quarterly 32.127 (1982): 127-136.

[4] Robinson, Robert B. "Classification of reservoir rocks by surface texture." AAPG Bulletin 50.3 (1966): 547-559.

[5] Maryland School Assessment Example. http://www.mdk12.org/assessments/k_8/items/sample_grade3_reading.html

[6] Cortese, Emma E. "The application of question-answer relationship strategies to pictures." The Reading Teacher 57.4 (2003): 374-380.

[7] Paris, Alison H., and Scott G. Paris. "Children's Comprehension of Narrative Picture Books. CIERA Report." (2001).

[8] van Kraayenoord, Christina E., and Scott G. Paris. "Story construction from a picture book: An assessment activity for young learners." Early Childhood Research Quarterly 11.1 (1996): 41-61.

[9] Summers-Stay, Douglas. *Machinamenta.* Machinamenta Press, 2011.

[10] Krathwohl, David R. "A revision of Bloom's taxonomy: An overview." Theory into practice 41.4 (2002): 212-218.

[11] Krathwohl, David R. Taxonomy of Educational Objectives: Cognitive domain. Vol. 1. Longmans, Green, 1956.

[12] Hertzmann, Aaron, et al. "Image analogies." Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM, 2001.

[13] Turk, Matthew, and Alex Pentland. "Eigenfaces for recognition." Journal of cognitive neuroscience 3.1 (1991): 71-86.

[14] Carson, Chad, et al. "Blobworld: Image segmentation using expectation-maximization and its application to image querying." Pattern Analysis and Machine Intelligence, IEEE Transactions on 24.8 (2002): 1026-1038.

[15] Fan, Jianping, Hangzai Luo, and Yuli Gao. "Learning the semantics of images by using unlabeled samples." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 2. IEEE, 2005.

[16] Boutell, Matthew R., Jiebo Luo, and Christopher M. Brown. "Scene parsing using region-based generative models." Multimedia, IEEE Transactions on 9.1 (2007): 136-146.

[17] Liu, Ce, Jenny Yuen, and Antonio Torralba. "Nonparametric scene parsing: Label transfer via dense scene alignment." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.

[18] Munoz, Daniel, J. Andrew Bagnell, and Martial Hebert. "Stacked hierarchical labeling." Computer VisionECCV 2010. Springer Berlin Heidelberg, 2010. 57-70.

[19] Jiang, Jiayan, and Zhuowen Tu. "Efficient scale space auto-context for image segmentation and labeling." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.

[20] W. Einhuser, M. Spain, and P. Perona, Objects predict fixations better than early saliency. Journal of Vision, 8(14):18, 126, 2008

[21] Foxe, JJ and GV Simpson. Flow of Activation from V1 to frontal cortex in humans. Experimental Brain Research, 2002.

[22] Mutch, J and DG Lowe. Multiclass object recognition with sparse, localized features. CVPR 2006

[23] Serre, T, L. Wolf and T. Poggio. Object recognition with features inspired by visual cortex. CVPR 2005.

[24] Miller, EK, CA Erickson and R Desimone. Neural mechanisms of visual working memory in prefrontal cortex of the macaque. Journal of Neuroscience, 1996.

[25] Desimone, R, TD Albright and CG Gross. Stimulus s selective properties of inferior temporal neurons in the macaque. Journal of Neuroscience, Vol 4, 1984.

[26] Weizmann Horse Database. http://www.msri.org/people/members/eranb/

[27] Kuettel, Daniel, and Vittorio Ferrari. "Figure-ground segmentation by transferring window masks." Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012.

[28] L. Bertelli, T. Yu, D. Vu, and S. Gokturk. Kernelized structural svm learning for supervised object segmentation. In CVPR, 2011

[29] Penn-Fudan Database for Pedestrian Detection and Segmentation. http://www.cis.upenn.edu/~jshi/ped_html/

[30] Seo, Hae Jong, and Peyman Milanfar. "Training-free, generic object detection using locally adaptive regression kernels." Pattern Analysis and Machine Intelligence, IEEE Transactions on 32.9 (2010): 1688-1704.

[31] Action Recognition of Static Images. http://www.cs.cmu.edu/~abhinavg/downloads.html

[32] Berkeley Segmentation Data Set and Benchmarks 500 http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

[33] M Reisenhuber, T Poggio. "Heirarchial models of object recognition in cortex." Nature Neuroscience 2, 1999

[34] Brox, Thomas, et al. "Nonparametric density estimation with adaptive, anisotropic kernels for human motion tracking." Human MotionUnderstanding, Modeling, Capture and Animation. Springer Berlin Heidelberg, 2007. 152-165.

[35] Summers-Stay, Douglas, and Yiannis Aloimonos. "Learning to recognize objects in images using anisotropic nonparametric kernels." Proc. 1st Annu. Meet. Biologically Inspired Cognitive Architectures (BICA) Society (eds Samsonovich AV, Jhannsdttir KR, Chella A., Goertzel B., editors.) (2010): 163-168.

[36] Laws, Kenneth Ivan. Textured Image Segmentation. No. USCIPI-940. UCLA Image Processing Inst., 1980.

[37] T. Ojala, M. Pietikinen, and D. Harwood (1994), "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions", Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994), vol. 1, pp. 582 - 585

[38] Pietikinen, Matti. Computer vision using local binary patterns. Vol. 40. Springer, 2011.

[39] Dror, Ron O., Edward H. Adelson, and Alan S. Willsky. "Recognition of surface reflectance properties from a single image under unknown real-world illumination." (2001).

[40] Liu, Ce, et al. "Exploring features in a bayesian framework for material recognition." Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010.

[41] Hu, Diane, Liefeng Bo, and Xiaofeng Ren. "Toward robust material recognition for everyday objects." Proc. BMVC. 2011.

[42] Bo, Liefeng, Xiaofeng Ren, and Dieter Fox. "Kernel descriptors for visual recognition." Advances in Neural Information Processing Systems 7 (2010).

[43] X. Ren and J. Malik. "Learning a classification model for segmentation." Proc. 9th Int. Conf. Computer Vision, volume 1, pages 10-17, 2003.

[44] Liu, Ming-Yu, et al. "Entropy rate superpixel segmentation." Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011.

[45] Brodatz texture dataset. http://www.ux.uis.no/~tranden/brodatz.html

[46] UIUC texture database. http://www-cvr.ai.uiuc.edu/ponce_grp/data/

[47] Flickr material database. http://people.csail.mit.edu/celiu/CVPR2010/FMD/

[48] Y. Rubner, C. Tomasi, and L. J. Guibas. "The Earth Movers distance as a metric for image retrieval." Technical Report STAN-CS-TN-98-86, Department of Computer Science, Stanford University, Sept. 1998.

[49] Pele, Ofir, and Michael Werman. "Fast and robust earth mover's distances." Computer vision, 2009 IEEE 12th international conference on. IEEE, 2009.

[50] Fast Earth Mover's Distance code. http://www.cs.huji.ac.il/~ofirpele/FastEMD/code/

[51] Rosenbaum, David A., et al. "The problem of serial order in behavior: Lashleys legacy." Human movement science 26.4 (2007): 525-554.

[52] Bauer, Patricia J. "Recalling past events: From infancy to early childhood." Annals of Child Development 11 (1995): 25-71.

[53] Whiten, Andrew, et al. "Imitation of hierarchical action structure by young children." Developmental science 9.6 (2006): 574-582.

[54] Fadiga, Luciano, et al. "Visuomotor neurons: Ambiguity of the discharge or motorperception?." International journal of psychophysiology 35.2 (2000): 165-177.

[55] Fogassi, Leonardo, et al. "Parietal lobe: from action organization to intention understanding." Science 308.5722 (2005): 662-667.

[56] Chomsky, Noam. Lectures on government and binding: The Pisa lectures. Vol. 9. Walter de Gruyter, 1993.

[57] Pastra, Katerina, and Yiannis Aloimonos. "The minimalist grammar of action." Philosophical Transactions of the Royal Society B: Biological Sciences 367.1585 (2012): 103-117.

[58] K.-C. Tai. The tree-to-tree correction problem. J. ACM. 1979

[59] K.Zhang and D.Shasha. Simple fast algorithms for the editing distance between trees and related problems. SIAM J. Comput. 1989.

[60] P.N. Klein. Computing the edit-distance between unrooted ordered trees. In European Symposium on Algorithms (ESA). 1998.

[61] M.Pawlik and N.Augsten. RTED: A Robust Algorithm for the Tree Edit Distance. PVLDB. 2011

[62] Eren Erdal Aksoy, Alexey Abramov, Johannes Dörr, Kejun Ning, Babette Dellen, and Florentin Wörgötter. Learning the semantics of object-action relations by observation. *International Journal of Robotics Research*, 30(10):1229–1249, 2011.

[63] P. Bauer. Recalling past events: from infancy to early childhood. *Annals of Child Development*, 11:25–71, 1995.

[64] R. Chalodhorn, D. Grimes, Rao R. Gabriel, and M. Asada. Learning humanoid motion dynamics through sensory-motor mapping in reduced dimensional spaces. In *ICRA*, 2006.

[65] Rizwan Chaudhry, Avinash Ravichandran, Gregory Hager, and René Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[66] Noam Chomsky. *Lectures on Government and Binding: The Pisa Lectures*. Mouton de Gruyter, 1993.

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

[68] L. Fadiga, L. Fogassi, V. Gallese, and G. Rizzolatti. Visuomotor neurons: ambiguity of the discharge or motor perception? *International Journal of Psychophysiology*, 35:165–177, 2000.

[69] Leonardo Fogassi, Pier Francesco Ferrari, Benno Gesierich, Stefano Rozzi, Fabian Chersi, and Giacomo Rizzolatti. Parietal lobe: From action organi-

zation to intention understanding. *Science*, 308:662 – 667, 2005.

[70] S. Gong and T. Xiang. Recognition of group activities using dynamic probabilistic networks. In *Proc. International Conference on Computer Vision*, 2003.

[71] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, 2007.

[72] Abhinav Gupta and Larry S. Davis. Objects in action: An approach for combining action understanding and object perception. In *CVPR*. IEEE Computer Society, 2007.

[73] S. Hongeng and R. Nevatia. Large-scale event detection using semi-hidden markov models. In *Proc. International Conference on Computer Vision*, 2003.

[74] Y. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

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

[76] H. Kjellstrom, J. Romero, and D. Kragic. Simultaneous visual recognition of manipulation actions and manipulated objects. In *Proc. European Conference on Computer Vision*, 2008.

[77] F. De la Torre, J. Hodgins, J. Montano, S. Valcarcel, R. Forcada, and J. Macey. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. Technical report, CMU-RI-TR-08-22, Robotics Institute, Carnegie Mellon University, July 2009.

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

[79] B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[80] Y. Li, C. Fermuller, Y. Aloimonos, and H. Ji. Learning shift-invariant sparse representation of actions. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[81] Ross Messing, Chris Pal, and Henry Kautz. Activity recognition using the velocity histories of tracked keypoints. In *ICCV '09: Proceedings of the Twelfth IEEE International Conference on Computer Vision*, Washington, DC, USA,

2009. IEEE Computer Society.

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

[83] D. Moore and I. Essa. Recognizing multitasked activities using stochastic context-free grammar from video. In *Proceedings of AAAI Conference,*, 2002.

[84] N. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. In *ICMI*, 2003.

[85] K Pastra and Y Aloimonos. The minimalist grammar of action. *Phil. Trans. R Soc. B*, 367(1585):103–117, 2012.

[86] C. Pinhanez and A. Bobick. Human action detection using pnf propagation of temporal constraints. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[87] David A Rosenbaum, Rajal G Cohen, Steven A Jax, Daniel J Weiss, and Robrecht Van Der Wel. The problem of serial order in behavior: Lashleys legacy. *Human Movement Science*, 26(4):525–554, 2007.

[88] M.S. Ryoo and J.K. Aggarwal. Recognition of composite human activities through context-free grammar based representation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[89] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto. Dynamic texture recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[90] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *ICPR*, 2004.

[91] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1-2):4–27, 2010.

[92] M. Sridhar, Anthony G. Cohn, and David C. Hogg. Learning functional object-categories from a relational spatio-temporal representation. In *Proc. 18th European Conference on Artificial Intelligence,*, pages 606–610, 2008.

[93] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008.

[94] I. Vicente, V. Kyrki, and D. Kragic. Action recognition and understanding

through motor primitives. *Advanced Robotics*, 21:1687–1707, 2007.

[95] J.M. Wang, D.J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[96] A. Whiten, E. Flynn, K. Brown, and T. Lee. Imitation of hierarchical action structure by young children. *Developmental Science*, 9:574–582, 2006.

[97] Geert Willems, Tinne Tuytelaars, and Luc J. Van Gool. An efficient dense and scaleinvariantspatio-temporal interest point detector. In *Proc. European Conference on Computer Vision*, 2008.

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

[99] Gooch, Bruce and Amy. Non-Photorealistic Rendering. A.K. Peters Ltd. 2001.

[100] Gardner, Howard. Artful scribbles: The significance of children's drawings. Basic Books (AZ), 1980.

[101] J. Willats. Art and Representation. Princeton U. Pr., 1997 page 199

[102] Durand, Frdo. "An invitation to discuss computer depiction." Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering. ACM, 2002.

[103] D. B. Lenat and R. V. Guha, Building large knowledge-based systems; representation and inference in the Cyc project. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989

[104] Sarjant, Samuel, et al. "All you can eat ontology-building: Feeding wikipedia to Cyc." Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01. IEEE Computer Society, 2009.