International Journal of Humanoid Robotics

RECOGNIZING BEHAVIOR IN HAND-EYE COORDINATION PATTERNS

WEILIE YI

Microsoft Corporation One Microsoft Way
Redmond, WA 98052, USA
weiliey@microsoft.com

DANA BALLARD

Department of Computer Science
University of Texas at Austin
Austin, TX, USA
dana@cs.utexas.edu

Received Day Month Year
Revised Day Month Year
Accepted Day Month Year

Modeling human behavior is important for the design of robots as well as human-computer interfaces that use humanoid avatars. Constructive models have been built, but they have not captured all of the detailed structure of human behavior such as the moment-to-moment deployment and coordination of hand, head and eye gaze used in complex tasks. We show how this data from human subjects performing a task can be used to program a dynamic Bayes network (DBN) which in turn can be used to recognize new performance instances. As a specific demonstration we show that the steps in a complex activity such as sandwich making can be recognized by a DBN in real time.

Keywords: Markov models; humanoid avatars ; dynamic Bayesian networks.

1. Introduction

Modeling human activity is central to a number of different areas in humanoid robotics. One of the most important is that of human-computer interfaces. In order to engage in a dialog with a human, a robot has to understand what the human is up to. The human conduct of everyday activities appears simple but has resisted robust solutions even though it has attracted considerable attention.[1,8,10,17–19] One reason may be that the role of evolution in the human design has been underestimated. The degrees of freedom of the human musculo-skeletal system, together with its control strategies make manipulation appear facile, but only because over evolutionary timescales hand-eye coordination strategies have been discovered and implicitly catalogued in the basic human design.[4] Thus it may be that we have to uncover these solutions by direct observation in order to be able to duplicate them to be

able to enter into a discourse about the human conduct in everyday tasks that acknowledges human spatio-temporal task constraints.[2, 15]

Researchers in psychology and physiology have generally reached a consensus that most of human vision system is active.[3, 7, 20] In the active vision paradigm, vision system is actively *seeks* information that is relevant to current cognitive activity.[16] The non-uniform spatial distribution of photoreceptors in the retina, which is just one observation point in a complex chain of reasoning, suggests that the vision system cannot, and doesn't have to, reconstruct the 3D model of the scene in the brain veridically, but it rather samples pieces of information from the scene by initiating eye movements and bringing the object of interest in the fovea. Because the high resolution retinal image being processed is extremely local to the fovea,[23] the process of any non-trivial visual action is a temporal sequence of both overt shifts of the fixation point[12] and internal movements of covert focus. Thus it is natural to model visual actions as sequences of small discrete fixations that have specific associated image processing goals and attendant functions.

Similarly, in hand-eye coordination tasks, in addition to the gaze control system, the rest of the the motor system also can be seen as composed of discrete steps, with each step generally coordinated with gaze shifts. Thus such behavior consists of high-speed segments, including the physical movement of the eyes, shifting of internal focus, memory manipulation, and other changes of internal states. Our hypothesis is that if we only keep 'snapshots' where important events happen, e.g. an object is recognized and remembered, or a point is fixated on, we can describe the visual behavior as a sequence of these events. If we further hypothesize that there is a finite, or even small number of these events, or basic operations, which are atomic, we have the basis for a visuo-motor routines theory. This theory, originally expressed for vision,[22] claims that there is a repository of precompiled programs in the brain, each of which could be triggerred on demand when a relevant task is present, and, these programs are assembled from a fixed set of primitive operations. The are assumed to be wired into brain's system and can be accessed and modified at run time. The visual routines theory models human visual perception in a top-down architecture: when a goal is identified, the corresponding routine is executed. The execution of visual routines is triggered by high level cognitive system, rather than low level stimuli. This model implies that cognition is situated: We cannot study visual cognition by extracting it from its context.

Understanding this internal structure is essential for behavior recognition. Human behavior recognition has been extensively studied and successively applied in various fields such as human computer interaction, context aware systems, security, transportation, and health care. It uses sensory data, e.g. video and motion tracker readings, to monitor and infer human activities.[5] We are particularly interested in recognizing behaviors in everyday hand-eye coordination tasks. Such tasks can involve extensive interactions with task relevant objects. Two prominent technical hurdles include searching for an object of interest in the scene from unsegmented low resolution images, and building a set of behavior models that can interpret

feature vectors computed from sensory input. We solve these two problems by detecting the attended object using eye tracking data without image segmentation, and by applying a probabilistic task model which can dramatically narrow down the choices in the inference of a sequence of activities.

Using eye gaze to recognize human attention is fairly common. Oliver[13] and Liu[9] investigated incorporating gaze analysis in human driver behavior modeling. Correlations between certain fixation patterns and driving tasks were found that could be used to help detect driver behavior and intention. However, few implementations in virtual reality simulations have been reported. The most closely related work to our system is by Yu and Ballard,[25] which recognized human actions in an office setting by parsing eye movement data and motion sensors data. The actions recognized involve only one object, and are typically short (less than 10 seconds) and isolated. Our system, using the same sensory input, can recognize complex behavior, e.g. making a sandwich, which is composed of a series of actions such as spreading and screwing, involving multiple objects. With the additional task model, we are able to integrate the task execution history into the inference system to enhance the efficiency of the recognition of a long sequence of actions. Philipose[14] also studied behavior recognition from interactions with multiple objects. Their approach is to build a recipe-like multi-step task description and determine the probability of each object being used in each step from online how-tos. Then human behavior is recognized by feeding a Bayesian inference engine the name of the object being used. Our task model is more generalized because it allows a task to be executed in different ways. For example, some steps can be done in arbitrary orders. This gives a much higher flexibility to the recognition system, which has the potential to recognize arbitrarily complex and loosely defined behavior.

This paper introduces a routine-based behavior modeling framework using a basic operation set which is task independent. The active vision approach is taken, and both visual and motor movements form the backbone of the interactions with the environment. This system concentrates on high level visual cognition and bypasses low level vision such as image segmentation and object recognition, assuming these processes are available for use. Because of the marriage between visual routines and active vision, it avoids the reconstruction of the surrounding's complete 3D model. Geometric information is retrieved on demand by executing proper routines, so the system can perform tasks in a geometry independent fashion.

The central component of the system is a Markov model which captures the interpersonal differences in solving natural tasks by executing a sequence of primitive operations. A task is automatically segmented into subtasks and the execution order is determined in a probabilistic fashion. In a very real sense the Markov model exhibits the ability to *anticipate* events because 'the past is prologue:' Having done a task many times allows us to experience its variations and provides us with the ability to do them differently as a function of different environmental and internal conditions. In addition the Markov model provides an ability to simulate the various ways of doing the task by choosing its state values incrementally in an online

manner.

The paper focuses on the task of making a sandwich. There are over 1,000 different ways to make a peanut butter and jelly sandwich, even after obvious symmetries are excluded. We use human data to develop a model in a two step process. First, hand segmented human data is automatically analyzed to find the common segments used by humans. Second, these segments are used to build a dynamic Bayes network model of the task.[11] The Bayes network makes extensive use of data gathered at the gaze fixation points to recognize the steps in the task in real time.

2. Recognizing sequences of visuo-motor primitives

Compared with deterministic computer agents, humans solve everday tasks in highly diverse ways. For instance, different people make sandwiches in different ways. It is important to identify the similarities and differences between those solutions, in the form of behavioral routines. We start with identifying local patterns in behavioral routines, and develope an algorithm to find common subsequences in different routines. The purpose is to partition sequential behavior into segments such that each behavior can be represented as an assembly of these segments in a certain order. Each behavior is a sequence of behavioral primitives and solves the same task. Since the actual planning in such a solution varies across individual subjects, this algorithm attempts to identify common structures and patterns in different solutions.

Since this algorithm assumes a routine-based modeling of cognitive behavior, complex behavior is programmed with a fixed set of primitives. For example, the following are two sample routines obtained by hand segmenting video data from different subjects making a sandwich. They encode structurally different while functionally similar behavior: Putting a slice of bread in a plate and using a knife to spread jelly on the bread. The routines are highly abstract and are for demonstration purpose only, so they do not encode detailed finger movement or visual feedback, and assume a proper context.

A closer look at these routines reveals that they share some common segments as shown below in Table 2. Each routines is a concatenation of the three segments. The only difference is the order. This small example is representative of much larger data sets used in making a complete sandwich. To compare these larger data sets we are motivated to construct an algorithm to identify these common segments automatically. To make the description of this algorithm more accessible. we use single letters for the segments. Thus the action 'locate bread' might be represented by the single character 'a.'

Since a behavior routine is coded as a text string, the automatic segmentation problem can be represented as finding a small common segmentation for a set of input strings, so that each string can be reconstructed by concatenating some or all of the segments in a certain order. By small we mean the number of segments is close to minimum. To simplify this problem, we assume the lengths of these strings

Routine 1	Routine 2
locate bread	locate knife
puthand right	puthand left
pickup right	pickup left
locate plate	locate bread
remember location_plate	puthand right
puthand right	pickup right
dropoff right	locate plate
locate knife	remember location_plate
puthand left	puthand right
pickup left	dropoff right
puthand left location_jelly	puthand left location_jelly
pickup left	pickup left
puthand left location_plate	puthand left location_plate
dropoff right	dropoff right

Table 1. Sample hand-segmented routines. This table shows two routines as the input to the automatic segmentation algorithm. These routines consist the same behavioral primitives with different order. They can be segmented into 3 segments, as hown in Table 2

Segment 1	Segment 2	Segment 3
locate bread	locate knife	puthand left location_jelly
puthand right	puthand left	pickup_left
pickup right	pickup left	puthand left location_plate
locate plate		dropoff_left
remember location_plate		
puthand right		
dropoff right		

Table 2. Finding common segments. The output of the automatic segmentation with input shown in Table 1

are equal, and all the segments appear in every input string exactly once.

The input to the algorithm is a series of strings

$$\{B^{(i)} | \, i = 1, 2, \ldots, N, B^{(i)} \in \Sigma^L, L > 0\}$$

where Σ is the alphabet and L is the length of the strings. And the output of the algorithm is

$$\{S^{(j)} | \, j = 1, 2, \ldots, M, \forall B^{(i)}, \exists p \in P(M) : B^{(i)} = S^{(p_1)} + S^{(p_2)} + \cdots + S^{(p_M)}\}$$

where P(M) is all the permutations of integers [1, M], and + is the string concatenation operator.

For example, if the input behavioral routines are

$B^{(1)}$	abcabbcabccbbbaccbabbccaba
$B^{(2)}$	ccbabcabbcabbccababbaccbab
$B^{(3)}$	bccabaabcabbcabccbbbaccbab

then the output consists of the four segments shown below. In reconstruction, those three input strings are corresponding to permutations 1234, 2143 and 4123 respectively. If the three input strings all solve the same task, then the four segments can be recognized as the *subtasks* of it, and the difference among various behaviors can be captured by the execution order of those subtasks.

$S^{(1)}$	abcabbcab
$S^{(2)}$	ccb
$S^{(3)}$	bbaccbab
$S^{(4)}$	bccaba

Obviously, every such problem has a trivial solution $S^{(j)} \subseteq \Sigma$: each segment is a character in the alphabet. A solution is meaningful in term of subtask recognition only if the number of segments is small. Even with a given number of segments, enumerating all the possible segmentations is computationally hard. If the length of the input strings is L, and the number of segments is M, there are

$$\binom{L}{M-1} = \frac{L!}{(L-M+1)!(M-1)!}$$

different segmentations.[a] The efficiency of the searching becomes very important. So we designed an algorithm with a greedy heuristic[6] that can produce a sub-optimal solution.

Our algorithm has two steps.

Step 1 Find all the potential segments. This step is an iterative search for all the common substrings of all input strings. We pick an arbitrary input string $B^{(i^*)}, i^* \in [1, N]$, compare it with another input string $B^{(i)}$, $i \neq i^*$, and collect all the segments in $B^{(i^*)}$ that also exist in $B^{(i)}$, by shifting and matching as shown in Fig 1. Here the location of such segments in $B^{(i^*)}$ is important: identical substrings in different locations are considered different segments. Repeat this process on all other input strings $B^{(i)}$, $i \neq i^*$, and compute the intersection of collected segments. This intersection has all the segments in $B^{(i)}$ that also exists in *all* other strings, i.e. the solution to the automatic segmentation problem consists of segments selected from this collection. We denote this candidate collection as

$$C = \{C_j | \, j = 1, 2, \ldots, K, C_j \subseteq [1, L]\}$$

each element of which is the location of a substring of $B^{(i^*)}$. Note if a substring c is in this candidate collection, then all its non-empty substrings are candidates too.

[a]For example, strings of 30 character can be segmented into 5 segments in 27405 different ways; strings 50 characters long can be segmented into 7 segments in 15890700 different ways.

Also note that the calculation is over potential strings, and that these may have repeated elements from the alphabet. Thus the calculation only counts the number of ways of dividing any string up, irrespective of alphabetic constitution.

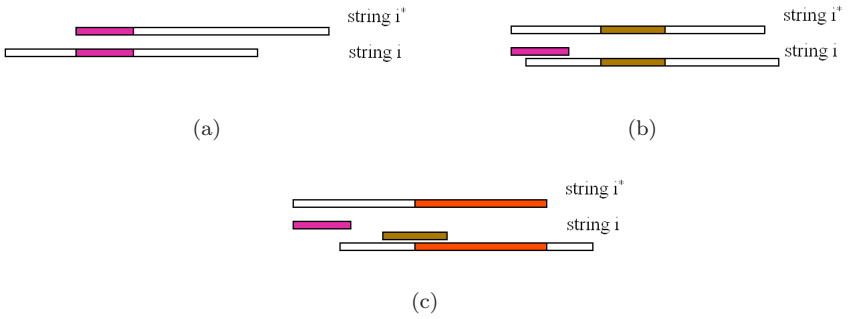


Fig. 1. Searching for all common substrings of a pair of input strings $B^{(i^*)}$ and $B^{(i)}$, $i \neq i^*$. (a)-(c): common substrings are indentified while string $B^{(i)}$ is shifts along $B^{(i^*)}$. Each of these common substrings are candidate segments. Any substrings of a candidate segment is also a candidate segment. For the sake of simplicity, they are not shown in this figure.

This step has a time complexity of $O(L^2N)$ in terms of comparisons, where N is the size of the alphabet..

Step 2 Search for a combination of candidate segments in C. The goal of this step is to find $S \subseteq C$ such that the substrings indexed by S is

$$\{S^{(j)}|j = 1, 2, \ldots, M, \forall B^{(i)}, \exists p \in P(M) : B^{(i)} = S^{(p_1)} + S^{(p_2)} + \cdots + S^{(p_M)}\}$$

which is the desired output. In general this is an NP Complete problem. Our approach is a knapsack algorithm which doesn't guarantee an optimal solution, but generates a sub-optimal one without having to exhaust all the combinations. The basic idea of this greedy algorithm is to give priority to long candidate segments when looking for a combination. First we sort C by length decreasingly, and get C^*. Then the algorithm is a depth first search of a binary decision tree in which the root nodes correspond to the decision "Is the longest candidate segment C_1^* in the segmentation?", and generally the ith layer nodes correspond to the decision "Is the ith longest candidate segment C_i^* in the segmentation?". As the depth first traverse proceeds, the combination of selected candidates, decided by the path from the root, as shown in Fig 2, is tested for the qualification for a solution to the problem, i.e. whether the current combination makes a segmentation of *each* input string $B^{(i)}$. Once this criterion is met, the search stops.

This algorithm doesn't guarantee an optimal solution and each run may generate different segmentation. Since the complexity of optimization is exponential in the

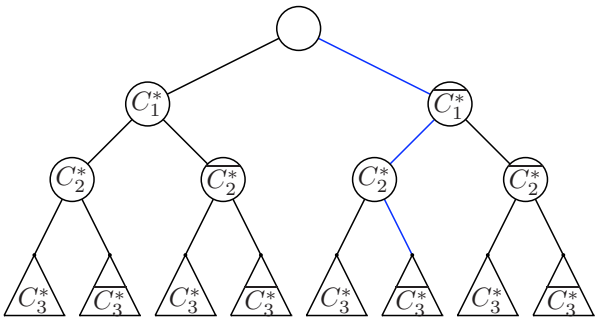


Fig. 2. Searching for a Segmentation. The decision tree has a height of $|C|$, with each layer asking the question whether segment C_i^* can be added in the segmentation given the decisions on segments $C_j^*, j < i$. The blue path indicates an example solution, in which segment C_2^* is included but C_1^* and C_3^* are not. The subtrees rooted at C_3^* and $\overline{C_3^*}$ are shown as triagles.

length of input strings, we use a greedy algorithm when dealing with long inputs and go back to brute force when input strings are short.

A few tricks accelerate the algorithm. For example, we compressed the input routines by encoding each primitive name with one letter, skipping all the separators. We tested this algorithm with visuo-motor routines describing a virtual human making sandwiches and the algorithm successfully identified ten segments, i.e. ten subtasks which can be solved with different orders. This segmentation, given in Table 3, is the basis of the next two sections.

3. Human Data

We recorded 4 human subjects making peanut butter and jelly sandwiches. The experimental setting is shown in Fig 3: objects are placed in the same positions as in the virtual reality. We used an SR Research EyeLink®II eye tracker to record gaze movement. We then analyzed the transitions of fixation points. One of the fixation patterns is illustrated in Fig 4(b).

Human eye movements are far more noisy and unpredictable than camera movement in a computer vision system. We processed human data by clustering fixation points and associating them to objects in the scene. Fortunately this association is easy: Almost every fixation point is close to a certain object which is relevant to the task. From the similarity between the two fixation pattern in Fig 4 we can see that the design of the instruction set in our behavioral routine architecture provides functional support for generating human like visual behavior. Furthermore, it has the extensibility and flexibility to model human world interaction in natural tasks, and provides a platform to study complex human visuo-motor behavior.

Table 3 summarizes the scheduling of 10 subtasks in making a peanut butter

Fig. 3. Experiment Setting. A subject is making a peanut butter sandwich with real objects which have identical positions as in the virtual reality.

sandwich by 3 human subjects[b]. Despite that some chronological constraints, e.g. BT, PLF and KH must precede POB and JOB, have ruled out most of the 10! orders, the number of possible orders remaining is still a large number[c]. However, experiments with additional subjects show that the orders picked by human subjects display common features.

4. Markov Model for Task Planning

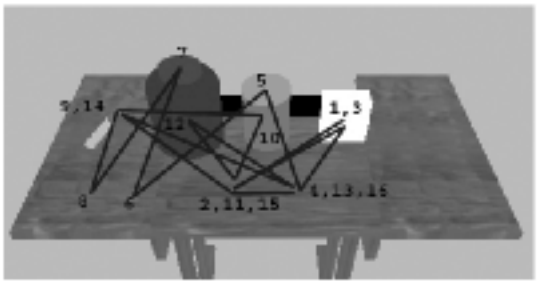
Different subjects have different orders in solving each subtask, but there are some statistically common patterns, e.g. KH is almost always followed by one of the two spreading subtasks, POB or JOB. This motivates us to use a Markov model to capture, and later generate, these patterns.

Generally, we denote these subtasks as

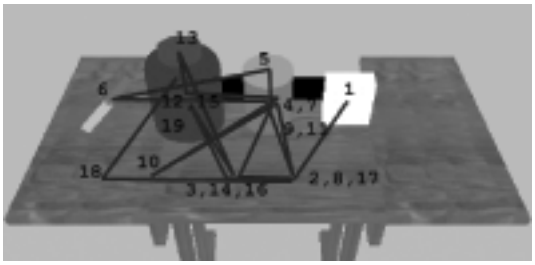
$$T = \{t_i | i = 1, 2, \ldots, N\}$$

[b]We make some assumptions such as the knife is picked up only once and is not put down until spreading finishes.

[c]If we divide the 10 subtasks into 3 stages: {BT, PLF, JLF, KH}, {POB, JOB} and {PLO, JLO, KT, FB}, we have at least $4! \times 2! \times 4! = 1152$ different orders



(a)



(b)

Fig. 4. Fixation Patterns in Sandwich Making. The numbers indicate order of fixations. Some points have multiple labels because of revisits. (a) is the fixation pattern of the virtual agent, which is controlled by handmade script, while (b) shows that of a human subject making a virtual sandwich in the same VR setting. Each fixation point is one of the following 10 positions. Upper row: knife, jelly bottle, jelly lid, peanut butter bottle, peanut butter lid. Lower row: the places to put jelly lid, peanut butter lid, left slice of bread, right slice of bread.

First we compute the power set of T:

$$P(T) = \{\emptyset, \{t_1\}, \{t_2\}, \ldots, \{t_1, t_2\}, \ldots, T\}$$

and then construct a graph in which the vertex set

$$V = P(T)$$

and edge set

$$E = \{\langle p_i, p_j \rangle | \, p_i, p_j \in V, p_i \subset p_j, |p_j - p_i| = 1\}$$

In plain words, each vertex of this graph is a set of subtask that have been done, indicating the progress in solving a complete task, while each edge is the transition between two states, one of which has exactly one more subtask done.

Since a subtask can be executed in various of scenarios, we need to encode the context of the subtask as well. For example, the subtask t_1 of the task set $\{t_1, t_2, t_3, t_4\}$ may be executed in 8 different scenarios, taking into account the precedence constraints.

SUBTASK LIST	SEQUENTIAL TIME INTERVALS									
	1	2	3	4	5	6	7	8	9	10
BT	abc									
PLF		a	c		b					
JLF		bc				a				
KH			ab	c						
POB				a	c	b				
JOB				b		c	a			
PLO					a				b	c
JLO									c	ab
KT							c	ab		
FB							b	c	a	

Table 3. Scheduling of Subtasks. The task is decomposed into ten subtasks including BT (putting bread on table), PLF (taking peanut butter lid off), JLF (taking jelly lid off), KH (grabbing knife in hand), POB (spreading peanut butter on bread), JOB (spreading jelly on bread), PLO (putting peanut butter lid on), JLO (putting jelly lid on), KT (putting knife on table), and FB (flipping bread to make an sandwich). Letters a, b and c denote the orders of subtasks taken by 3 subjects, e.g. in the first 2 steps subject c put bread on the table and took jelly lid off.

$$\begin{aligned}
\emptyset &\rightsquigarrow \{t_1\} \\
\{t_2\} &\rightsquigarrow \{t_1, t_2\} \\
\{t_3\} &\rightsquigarrow \{t_1, t_3\} \\
\{t_4\} &\rightsquigarrow \{t_1, t_4\} \\
\{t_2, t_3\} &\rightsquigarrow \{t_1, t_2, t_3\} \\
\{t_2, t_4\} &\rightsquigarrow \{t_1, t_2, t_4\} \\
\{t_3, t_4\} &\rightsquigarrow \{t_1, t_3, t_4\} \\
\{t_2, t_3, t_4\} &\rightsquigarrow \{t_1, t_2, t_3, t_4\}
\end{aligned}$$

Each of the 8 scenarios, although executing the same subtask, should be different states. To derive the new task model, we start from the original graph, add a node to each edge, and removed the original nodes. For example, the following transitions

$$\ldots \{t_1\} \xrightarrow{(t_2,p)} \{t_1, t_2\} \xrightarrow{(t_4,q)} \{t_1, t_2, t_4\}$$

become

$$\ldots \{t_1\} \xrightarrow{p} \big(\{t_1\}, \{t_1, t_2\}\big) \xrightarrow{1} \{t_1, t_2\} \xrightarrow{q} \big(\{t_1, t_2\}, \{t_1, t_2, t_4\}\big) \xrightarrow{1} \{t_1, t_2, t_4\}$$

after inserting nodes on the edges. p and q are transitional probabilities. The new nodes are labeled as pairs of sets of subtasks, indicating the context of the subtask being executed. These transitions then become

$$\big(\emptyset, \{t_1\}\big) \xrightarrow{p} \big(\{t_1\}, \{t_1, t_2\}\big) \xrightarrow{q} \big(\{t_1, t_2\}, \{t_1, t_2, t_4\}\big)$$

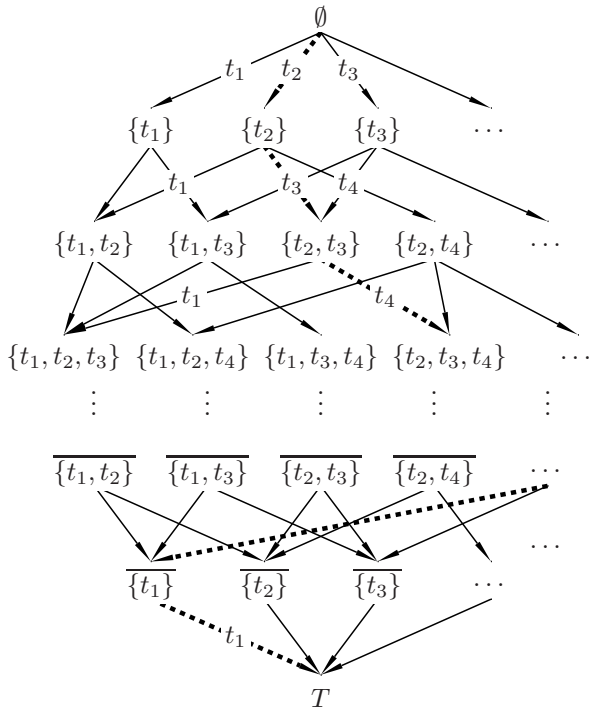


Fig. 5. The Markovian task model represents knowledge about task planning extracted from human experiments. Each node is a set of accomplished subtasks, which is a state in solving the task. Each edge is a subtask that brings task solving one step closer to the end state.

after removing the original nodes. Transitional probabilities are inherited by new edges. This new representation makes explicit which subtask is being executed with what context, in each state. These states are the states of the *task* node.

The *subtask* node has much less states, which have a one to one correspondence with all the subtasks. The probabilistic dependency between the *task* node and the *subtask* node is straight forward: there is an m-to-1 mapping between the two. For example, all the 8 different edges listed above map to subtask t_1, no matter what the context is. So the conditional probabilistic distribution matrix of node *subtask* is binary: each row is all 0's except one element, which is 1.

With this new representation of the task model, the time frame of each state is limited to a much smaller range. Since the contextual information, i.e. which subtasks have been executed, is known, the system doesn't have to consider all the scenarios. Instead, it only has to process the subtasks that have actually been executed. Without the max and min operators in Eq. 4, we have a better estimation of the likelihood of observing a clock reading given the current state.

After collecting human data and assigning values to transitional probabilities

in the task model, behavior recognition can be computed in a much more efficient way. These probabilities restrain the paths through time slices in the DBN. For example, the subtask execution order $\vdash t_1 \rightarrow t_5$ is unlikely to be taken if the transitional probability between states $\{\emptyset, \{t_1\}\}$ and $\{\{t_1\}, \{t_1, t_5\}\}$ is close to zero. These constraints can greatly reduce ambiguity in the inference.

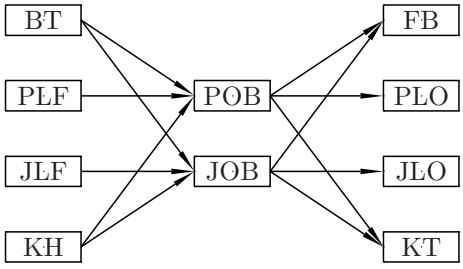


Fig. 6. Dependencies among Subtasks. Arrows connecting subtask t_i and t_j denotes that t_i must be done prior to t_j.

Note that in the construction of such a DAG, constraints about chronological order of subtasks apply, as shown in Fig 6. For example, POB always happens after PLF, and before FB. With these constraints, the number of vertices is narrowed down from 2^{10} to 41.

At this point we have a general framework for learning a task model from human data based on.[24] In this framework, a natural task is automatically segmented into several subtasks, providing a way to represent and measure task progress. The task execution becomes a navigation through a series of states, gradually leading to the end state. The transitional probabilities between these states are represented by a Markovian task model, as shown in Fig. 11, which computes the probabilities from human data. The model represents knowledge about task planning extracted from human experiments. Each node is a set of accomplished subtasks, which is a state in executing the task. Each edge is a subtask that brings the task one step closer to the end state $T = \{t_1, t_2, \ldots, t_N\}$. Any path from $\emptyset$ to T gives an order of executing the N subtasks, while each N hops from $\emptyset$ always brings the task to its end state T. More details are in.[24]

Next, we feed data into this Markov model, and calculate transition probabilities between statuses. With these probabilities, subtask orders can be generated probabilistically. An example is as follows.

With input data (numbers indicating subtasks):

Subject 1	0 3 1 4 7 2 5 6 9 8
Subject 2	2 0 1 3 4 5 9 6 8 7
Subject 3	0 2 3 5 1 4 6 9 7 8
Subject 4	0 2 1 3 4 5 9 6 8 7

the most probable path is:

(75%) 0 → (67%) 2 → (67%) 1 → (100%) 3 → (100%) 4 → (100%)
5 → (67%) 9 → (100%) 6 → (67%) 8 → (100%) 7

with a probability 14.8%.

5. Incorporating Sensory Data

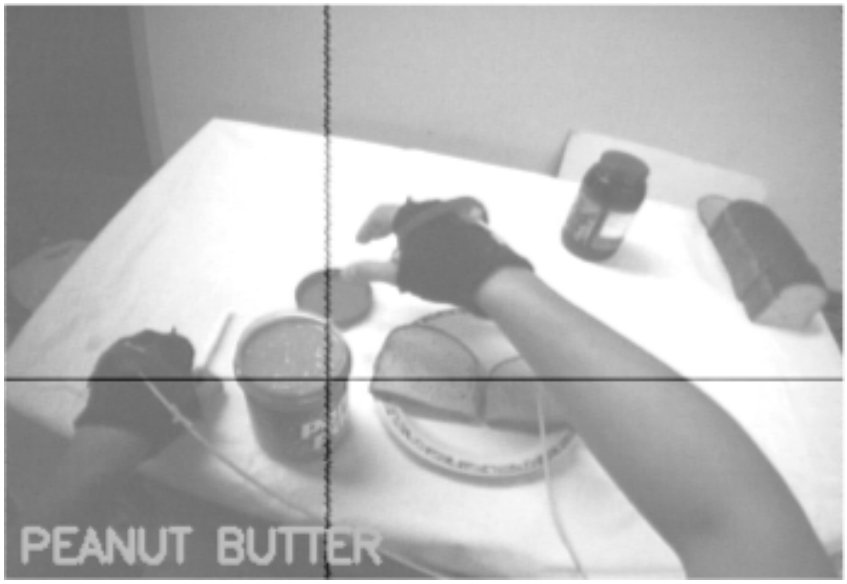
At this point tasks can be identified as sequences of visuo-motor primitives that are repeated in different data sets from different subjects. Furthermore, the tasks graphical data structure allows the different ways of making a sandwich to be characterized as paths in the graph. At this point we can add the methodology for taking task-related measurements from image and motor data. The data in the system includes the visually attended object, hand movements, and the current clock reading.

5.1. *Visually Attended Object*

An agent has intensive interactions with objects throughout a natural task. Which object the agent is interacting with is a very important piece of information needed in recognizing the agent's behavior. One approach to getting this information is to attach a sensor to each task relevant object.[14] However, this might not be possible for some objects, such as a slice of bread, which could be consumed by the agent. Instead, we use a head mounted camera and an eye tracker to recognize attended object from video.

Foveal Image Human makes a series of eye movements when inspecting a scene or searching for an object.[23] Along with the foveal design of the eye, the eye movement reduces the information to be processed in each fixation. Similarly, the use of eye tracker greatly reduces the difficulty of retrieving useful information from the scene. Image segmentation is no longer necessary because the image patch to be processed is dramatically narrowed down to a small area that projects to the fovea, where the visual acuity is the highest on the retina.

In human visual system, the fovea covers about 1.7 degree of visual field. Taking into account some peripheral vision and compensating for calibration errors in the eye tracker, we used a 12.65 degree visual field to recognize attended object. This corresponds to an image patch of 20 pixels in radius.



(a)

(b)

Fig. 7. Complete vs. foveated image. The behavior recognition does not use the entire image (a) but only makes use of a small image patch centered on the fixation point (b). The color histogram data in this patch is used for classification into one of $\{nothing, peanut_butter, jelly, bread, hand\}$

Color Histogram Since the image patch to be processed is small and normally the background does not constitute a major portion of the image patch, we used histogram intersection technique[21] to recognize attended objects. Histogram Intersection is defined as

$$X^o = \sum_{j=1}^{n} \min(I_j, M_j^o) \tag{1}$$

where M_j^o are all the models of object o.

In our system, histogram is computed in HSV color space, using saturation and hue channels. The match between the image and the model is

$$H(I, M^o) = \frac{\sum_{j=1}^{n} \min(I_j, M_j^o)}{\sum_{j=1}^{n} M_j^o} \tag{2}$$

The object with the highest match is the object being fixated.

$$x = \arg\max_{o \in O} H(I, M^o) \tag{3}$$

With histogram intersection, we do not need to segment the objects from background, nor do we have to maintain 3D models of the objects. In our sandwich making experiments, the object recognition program recognizes 4 different objects: *bread*, *peanut butter*, *jelly* and *hand*[d] with a histogram database of 30 models. The result of recognition is shown in Fig. 8.

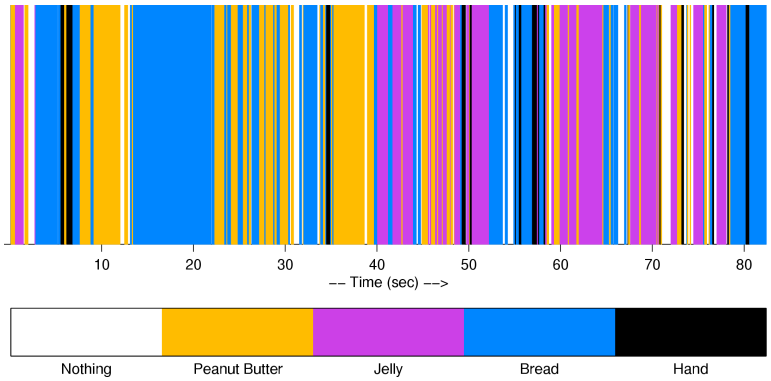


Fig. 8. Visually Attended Object. The upper bar shows what object is identified at each time interval. The lower bar illustrates the color coding key for the four objects.Close-set multiple colors are likely classification errors as the gaze typically uses 100-300 milliseconds per fixation. However this data can still be very useful when combined with prior expectations.

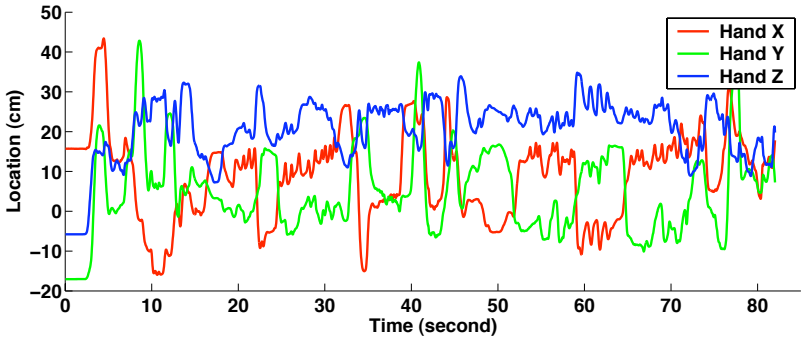
5.2. *Hand Movement Extraction*

Hand movement is another piece of important information that helps behavior recognition. Because of the highly dynamic and unpredictable nature of eye movement, especially look-aheads, visually attended object may not be the object being manipulated by hand. So directly recognizing hand movement becomes inevitable. We identified two events in our Bayesian inference system: reaching and high frequency hand movement. These two types of movement are commonly seen in natural tasks, and they characterize how the hands are interacting with objects.

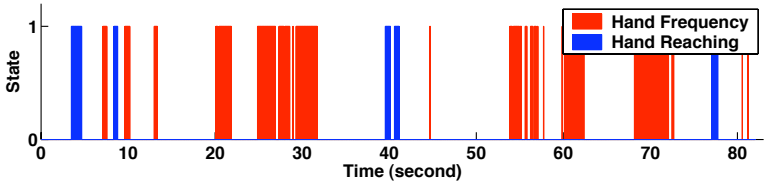
Hand locations are captured by a Polhemus FASTRAK^{TM}sensor attached to a pair of gloves worn by human subjects. These sensors produce 3D locations of the hands at a frequency of 120 Hz, with an accuracy of 0.01 inch (Fig. 9(a)).

[d]There are other objects in the task, for example, knife and table. We do not need to recognize every object because attended object is just one of a few observed nodes in the Dynamic Bayesian Network. Other other observable sensory data, along with the task model, help make robust inference from imperfect observations.

Reaching often accompanies picking up and putting down an object. The reaching event is defined as the distance from the hand to the body exceeding a threshold, which is 40 cm in our system for sandwich making. Depending on this distance, the corresponding observed node in DBN has two possible states: *reaching* and *not reaching*.



(a) Raw hand location data for the three wrist coordinates.



(b) Observed hand movement classifications

Fig. 9. Hand Movement Extraction. Certain places in the hand trajectory have recognizable features. Places where the velocity is high can be classified as 'Hand Reaching.' places with steady oscillations in one or more components can be classified as 'Hand Frequency,' a feature used to identify screwing or unscrewing container lids.

High frequency hand movements are typically seen in unscrewing a cap, stirring coffee, or spreading butter. We recognize them by computing the frequency of change of direction in hand movements. This frequency is computed in each of the 3 coordinates, and the maximal value is used as the main frequency. For right handed subjects, the left hand is mainly used to hold objects during unscrewing, stirring and spreading, so we only take into account the right hand's data for right handed subjects. Similar to the reaching event, the observed node for hight frequency hand movement has two states, separated by a threshold of 2/3 Hz.

Fig. 9(b) shows the observed hand movement from raw sensory data in Fig. 9(a). *Hand Reaching* has two states, *reaching* (1) and *not reaching* (0). *Hand Frequency* also has two states, *High frequency movement* (1) and *Low frequency movement* (0).

5.3. *Timing*

When a subtask can be executed depends on task planning, which is subject to precedence constraints between subtasks. For example, in sandwich making, taking jelly lid off has to precede, although not necessarily immediately, spreading jelly on bread. If the duration of taking the lid off is d seconds, then the earliest time that the subtask spreading jelly can start is d. Generally, the earliest time a task t can start is the sum of the duration of all its prerequisite subtasks, and the latest time t has to finish is the the total duration of the task minus the sum of the duration of all its postrequisite subtasks. Since the duration of each subtask is a probabilistic distribution, rather than a constant value, the time frame of a subtask t is

$$TF(t) = \Big[\sum_{i \in pre(t)} \min D(i), \sum_{t \notin post(t)} \max D(i) \Big] \tag{4}$$

This gives an additional constraint to the behavior recognition. For example, a high frequency hand movement in the first few seconds of the task is not likely to be spreading jelly, but more likely to be unscrewing a lid.

A stronger constraint will be introduced in the next section, taking advantage of human data.

6. Behavior Recognition

With all the sensory data discussed above, we are ready to design the DBN that inferences underlying task status from observations. The structure of the network is shown in Fig. 6.

There are 4 hidden nodes and 4 observed nodes in each time slice. As illustrated in Fig. 6The observed nodes are *recognized attended object*, *hand reaching*, *high frequency hand movement* and *time frame*. The hidden nodes are *gaze object*, *hand object*, *subtask* and *task*. *hand object* is the object being manipulated by hand; while *gaze object* is the object being fixated by eyes. They could be different because, (1) gaze often switches to a new object earlier than hand to guide subsequent hand movement; and (2) gaze may move way from the hand object during look-aheads. The probabilistic distribution matrix of the *gaze object* node reflects this relation statistically. *hand object* depends on *subtask* in the way that each subtask has a set of task relevant objects that are to be manipulated. The reason that we have two nodes, including *task*, to represent task progress, is that the original Markovian task model does not fit in the DBN.

Task planning information provided by the task model greatly narrows down the possible states that the agent could be in, given a series of observations.

In the previous section we were able to generate possible sandwich construction sequences, but recognizing these sequences in real-time is harder. The goal is to compute an agent's action from observed events in the sensori-motor input. A Bayes network is a suitable tool for this class of problems because it uses easily observable

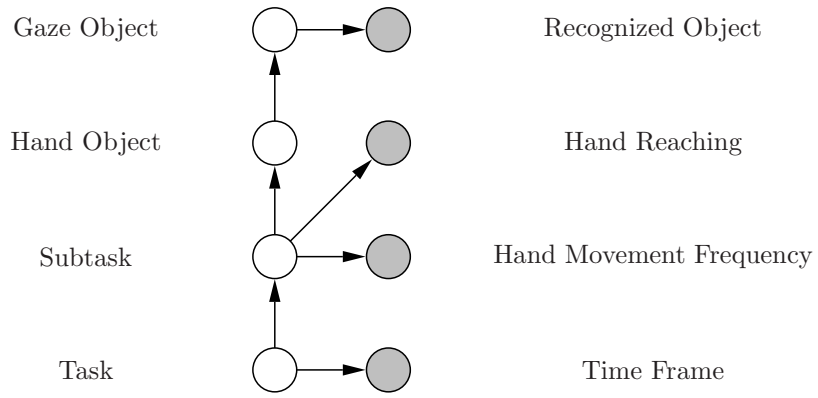


Fig. 10. Structure of the Dynamic Bayes Net. There are four hidden nodes, which denotes the internal states of an agent, and four observable nodes, which are sensory data retrieved from human object interactions in a natural task.

evidence to update or to newly infer the probabilistic distribution of the underlying random variable. A Bayesian net represents the causalities with a directed acyclic graph, its denoting variables and edges denoting causal relations.

Since the state of the agent is dynamically changing, and the observations are being updated throughout the task solving process, we need to specify the temporal evolution of the network. Fig. 6 illustrates the two slice representation of a Dynamic Bayesian Network. A DBN is a extension of Hidden Markov Models, in that a DBN can have multiple hidden and observed states which have complex interdependencies.

The two slice representation can be easily unrolled to address behaviors with arbitrary numbers of slices. In each time, the observed sensory data (gray nodes), along with its history, are used to compute the probability of the hidden nodes being in certain states:

$$P(\boldsymbol{Q}^t|\boldsymbol{O}^{[1,t]})$$

where $\boldsymbol{Q}^t$ is the set of states of hidden nodes at time t, $\boldsymbol{O}^{[1,t]}$ are the observations over time span $[1,t]$, and

$$P(\boldsymbol{Q}^t|\boldsymbol{O}^{[1,t]}) = P(\boldsymbol{Q}^1)P(\boldsymbol{O}^1|\boldsymbol{Q}^1)\prod_{t=2}^{t} P(\boldsymbol{Q}^t|\boldsymbol{Q}^{t-1})P(\boldsymbol{O}^t|\boldsymbol{Q}^t) \tag{5}$$

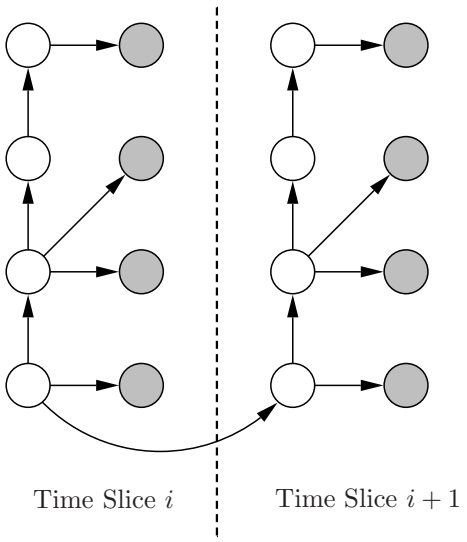


Fig. 11. Two Slice Representation of the Dynamic Bayes Net. Shaded nodes are observables; the others are hidden. Causalities, represented by straight arrows, are determined by probability distribution matrices. The state of the lowest hidden node is determined by its prior distribution in the first time/slice, and thereafter jointly determined by its previous state and the transition matrix, as denoted by the curved arrow.

Behavior recognition is to compute the states of each hidden node $\boldsymbol{S}^t$ at time t that maximize the probability of observing the given sensory data:

$$\boldsymbol{S}^t = \arg\max_{\boldsymbol{S}} P(\boldsymbol{Q}^t = \boldsymbol{S}|\boldsymbol{O}^{[1,t]}) \tag{6}$$

Eq. 6 defines the basic algorithm of the recognition system. The sensory data we used in our system include visually attended object, hand movement and the process time. The following sections describe the structure of the Dynamic Bayesian Network and details about data preprocessing and behavior inferencing.

7. Behavior Recognition Experiments

We tested our system in the peanut butter & jelly sandwich experiment. there are 4 objects involved: bread, peanut butter, jelly and hand. Hand is included because it is often fixated by the eyes. Identifying exactly what is being fixated helps inference what subtask is being executed, since the probability of fixating the hand could be different across all subtasks. Both the hidden *gaze object* node and the observed *recognized object* node have 5 states, one for each object, and another for *nothing*. Similarly, the *hand object* has 4 states, counting out the *hand* object.

Table 4. Hidden Nodes in the PBJ DBN.

Node Name	Number of States
task	80
subtask	10
hand object	4
gaze object	5

Table 5. Observed Nodes in the PBJ DBN.

Node Name	Number of States
time frame	20
hand frequency	2
hand reaching	2
recognized object	5

The *gaze object* node could be in *nothing* state during a saccade, or when the gaze is distracted by a task irrelevant stimulus.

The *recognized object* node is in *nothing* state when the eye tracker fails to track the fixation point (e.g. when the subject is blinking), or the computer vision module is not able to recognize any object, or what is recognized is a background object which the hand does not manipulate, for instance the table or the plate. To simplify the model and speed up the inference, we did not include these two objects, although they can be recognized.

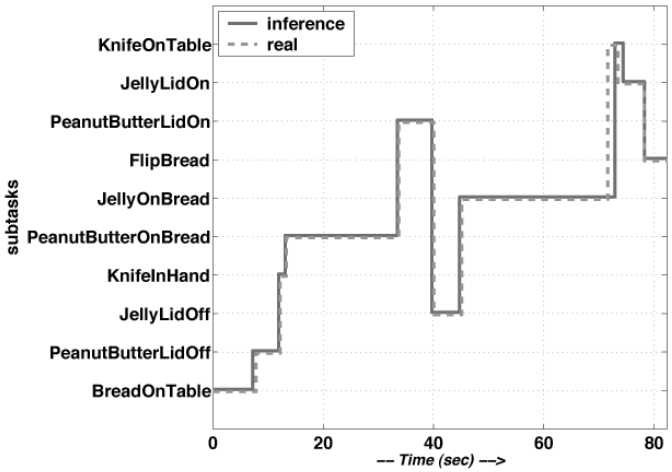
A task relevant object that we cannot recognize is the knife. Its slim profile makes histogram intersection a non-optimal algorithm to use. However, with other constraints, we can also pinpoint the moment when a knife is being picked up.

The *time frame* node has 20 states. Each state corresponds to a 5 second interval, and the maximal duration of a sandwich making task is 100 seconds. The granularity can be tuned. A finer grained DBN gives more accurate tracking but can also make the inference slower than a coarser one.

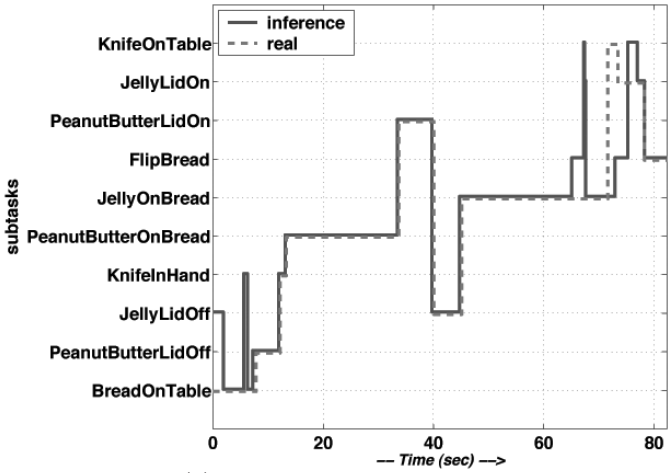
The results of the recognition are shown in Fig. 12. There are two recognition modes. In the offline mode, the task state at each moment is inferred after observing the *complete* set of observation data, i.e. $P(\boldsymbol{Q}^t|\boldsymbol{O}^{[1,T]})$, where $t \leq T$ and T is the duration of the task. In the online mode, the inference is only based on what has happened, at each moment. In either modes, the system recognizes subtasks correctly most of the time.

8. Discussion and Conclusions

Inspired by the Visual Routines model, we developed a computer vision architecture which takes an active and situated approach, focuses on high level vision, and



(a) Offline behavior recognition



(b) Online behavior recognition

Fig. 12. Recognition Results.

exploits embodied information. Unlike any previous work, it has a basic operation set with which virtually any high level visuo-motor behavior can be constructed, and has the potential to model complex human world interaction as programming with perceptual and motor primitives.

Based on the sequential modeling of cognition and agent-world interaction, we introduced an algorithm to automatically identify top level structure of behavioral routines by finding common segmentations across different routines, and dividing a task into subtasks.

Our experiments show that humans make the same kinds of trade-offs that are made in the visual routines model. In coffee pouring, individual subjects stop at

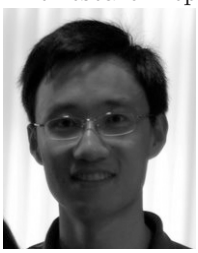
a fixed distance from the top of the cup and have very small deviations between the level in successive fills, suggesting the use of standard procedures. Also, as shown by our model, most of this variance can be explained as noise in an image matching process. In sandwich-making, individual subjects exhibit very similar task orderings so that their performance can be easily captured by a behavioral routines based Markov Decision Process that contains the alternate task orderings.

We also described a Markov model which can capture different orders in which human subjects execute subtasks. This model can be used to generate behavioral routines that solve the same task in a human like fashion.

As future work, we are interested in a more robust automatic segmentation algorithm. Currently we can only process very regular behaviors, i.e. corresponding segements in different behaviors must be identical. However, the same subtask could be executed in slightly different ways. A soft comparison criteria is necessary to address this problem.

Acknowledgments

The research reported herein is supported by NIH Grant R01 RR009283



Weilie Yi received his/her M.S. degree in Computer Science from Fudan University in Shanghai, China, and his Ph.D. degree in Computer Science from the University of Rochester, USA, in 2001 and 2006, respectively. From 2006, he has been at at Microsoft Research. At Microsoft, Yi works as a Software Design Engineer, working on a variety of products including Natural User Interface, Digital Ink Analysis, and Search Relevance. Yis research interests include human computer interaction, cognitive science, machine learning, and information retrieval.

Dana Ballard received his/her M.S. degree in Information and Control Engineering from the University of Michigan, USA, and his Ph.D. degree in Information Engineering from the University of California,Irvine, USA, in 1970 and 1974, respectively. From 1975 to 2005, he was at the University of Rochester. He/she is currently a Full Professor in the Department of Computer Science, the University of Texas at Austin. Ballard's main research interest is in computational theories of the brain with emphasis on human vision. In 1985 he and Chris Brown led a team that designed and built a high speed binocular camera control system capable of simulating human eye movements and mounted on a robotic arm that allowed it to move at one meter/second in a two-meter-radius workspace. Ballard and Brown also wrote the first Computer Vision text.

References

1. Tamim Asfour, Florian Gyarfas, Pedram Azad, and Rudiger Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. In *Humanoids*, 2006.
2. N. Badler. Virtual beings. *Communications of the ACM*, 44:33–35, 2001.
3. Dana H. Ballard. Animate vision. *Artificial Intelligence*, 48:57–86, 1991.
4. Eric B. Baum. *What is Thought?* MIT Press, 2004.
5. Cynthia Breazeal, Daphna Buchsbaum, Jesse Gray, David Gatenby, and Bruce Blumberg. Learning from and about others: Towards using imitation to bootstrap the social understanding of others by robots. *Artificial Life*, 11:31–62, 2005.
6. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.
7. John M. Findlay and Iain D. Gilchrist. *Active Vision: The Psychology of Looking and Seeing*. Oxford University Press, 2003.
8. Odest Chadwicke Jenkins, German Gonzalez Serrano, and Matthew M. Loper. Interactive human pose and action recognition using dynamic motion primitives. *International Journal of Humnoid Robotics*, 4:365–385, 2007.
9. Andrew Liu and Salvucci. Modeling and prediction of human driver behavior. In *Proc. 9th HCI International Conference*, pages 1479–1483, 2001.
10. M. Lopes and J. Santos-Victor. Visual learning by imitation with motor representations. *IEEE Transactions on Systems, Man and Cybernetics*, 35:438–449, 2005.
11. Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California at Berkeley, 1994.
12. David Noton and Lawrence Stark. Scanpaths in eye movements during pattern perception. *Science*, 171(3968):308–311, 1971.
13. Nuria Oliver. *Towards Perceptual Intelligence: Statistical Modeling of Human Individual and Interactive Behaviots*. PhD thesis, MIT, 2000.
14. Matthai Philipose, Donald J. Patterson, Dieter Fox, Henry Kautz, and Dirk Hahnel. Inferring activities from interactions with objects. In *Pervasive Computing, IEEE*, volume 3, pages 50–57, 2004.
15. C. Phillips, J. Zhao, and N. Badler. Interactive real-time articulated figure manipulation using multiple kinematic constraints. *Computer Graphics*, 24:245–250, 1990.
16. C. A. Rothkopf, D. H. Ballard, and M. M. Hayhoe. Task and context determine where you look. *Journal of Vision*, 7:1–20, 2007.
17. Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3:233–242, 1999.
18. Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society B*, 358:537–547, 2003.
19. A. P. Shon, J. J. Storz, A. N. Meltzoff, and R. P. N. Rao. A cognitive model of imitative development in humans and machines. *International Journal of Humnoid Robotics*, 4:387–406, 2007.
20. N Sprague, D Ballard, and Al Robinson. Modeling embodied visual behaviors. *ACM Transactions on Applied Perception*, 4, 2007.
21. Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
22. Shimon Ullman. *High-level vision*. MIT Press, 1996.
23. Alfred L. Yarbus. *Eye movements and vision*. New York, Plenum Press, 1967.
24. Weilie Yi and Dana H. Ballard. Routine based models of anticipation in natural behaviors. In *AAAI Fall Symposium, From Reactive to Anticipatory Cognitive Embodied Systems*, pages 141–147, Arlington, VA, November 2005.

25. Chen Yu and Dana H. Ballard. Learning to recognize human action sequences. In *Proc. IEEE International Conference on Development and Learning*, Cambridge, MA, June 2002.