M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 376
To appear in AAAI'96 – Portland, Oregon, 1996

Approximate World Models: Incorporating Qualitative and Linguistic Information into Vision Systems

Claudio S. Pinhanez and **Aaron F. Bobick**
pinhanez — bobick@media.mit.edu

Abstract

Approximate world models are coarse descriptions of the elements of a scene, and are intended to be used in the selection and control of vision routines in a vision system. In this paper we present a control architecture in which the approximate models represent the complex relationships among the objects in the world, allowing the vision routines to be situation or context specific. Moreover, because of their reduced accuracy requirements, approximate world models can employ qualitative information such as those provided by linguistic descriptions of the scene. The concept is demonstrated in the development of automatic cameras for a TV studio – *SmartCams*. Results are shown where SmartCams use vision processing of real imagery and information written in the script of a TV show to achieve TV-quality framing.

1 Introduction

It has been argued — e.g. in [12] — that in any given situation most visual tasks can be performed by a relatively simple visual routine. For example: finding the ground reduces to finding a large (body-relative) horizontal plane if the observer is vertical and there are no other large horizontal planes. The difficulty in general, of course, is how to know the current state of the world without having to do all the detailed visual tasks first. The numerous possibilities for the fundamental relationships between objects in the scene is as much responsible for the complexity of vision as is the difficulty of the visual routines themselves.

The goal of this paper is to argue that vision systems should separate these two sources of complexity by using coarse models of the objects in the scene called *approximate world models*. This proposal is based on the observation that the real world does not need to be fully and accurately understood to detect many situations where a specific vision method is likely to succeed or fail. For instance, full and precise 3D reconstruction of the human body is not necessary to detect occlusion if the objective of the system is just to recognize faces of people walking through a gate.

A main feature of approximate world models is that their imprecision facilitates the use of incomplete and inaccurate sources of information such as linguistic descriptions of the elements and actions in a scene. In fact, we show in this paper that linguistic information can play a pivotal role in providing the contextual information needed to simplify the vision tasks.

We are employing approximate world models in the development of *SmartCams*, automatic TV cameras able to frame subjects and objects in a studio according to the verbal requests from the TV director. Our SmartCams are tested in the domain of a cooking show. The script of the show is available to SmartCams (in a particular format), and the cameras are shown to be able to produce TV-quality framing of subjects and objects.

2 Approximate World Models

Approximate world models are coarse descriptions of the main elements of a scene to be used in the selection and control of vision routines. These models are to be incorporated into vision-based systems built from a collection of different, simple, task-specific vision routines whose application is controlled according to the conditions described by the approximate world models.

This proposal comes from the observation that a common reason for the failure of vision routines — especially, view-based methods — is related to the complex geometric relationships among objects in the world. For example, often template-based tracking routines produce wrong results due to partial occlusion. In such situations, a crude 3-D reconstruction of the main objects in the scene can determine if the tracked object is in a configuration where occlusion is probable.

The advantages of using approximate models are at least three-fold. First, coarse reconstruction of the 3-D is arguably within the grasp of current computer vision capabilities. Second, as shown in this paper, control of task-specific vision routines can be based on inaccurate and incomplete information. And third, as we will demonstrate, reducing the accuracy requirements enables the use of qualitative information which might be available to the vision system.

Coarse and/or hierarchical descriptions have been used before in computer vision [3, 6]. Particularly, Bobick and Bolles employed a multi-level representational system where different queries were answered by different representations of the same object. Part of the novelty of our work is related to the use of the models in the dynamic selection of appropriate vision methods according to the world situation. And compared to other architectures for context-based vision systems, like Strat and Fischler's *Condor* system, [12], approximate world models provide a much more clear distinction between the vision component and the 3-D world component.

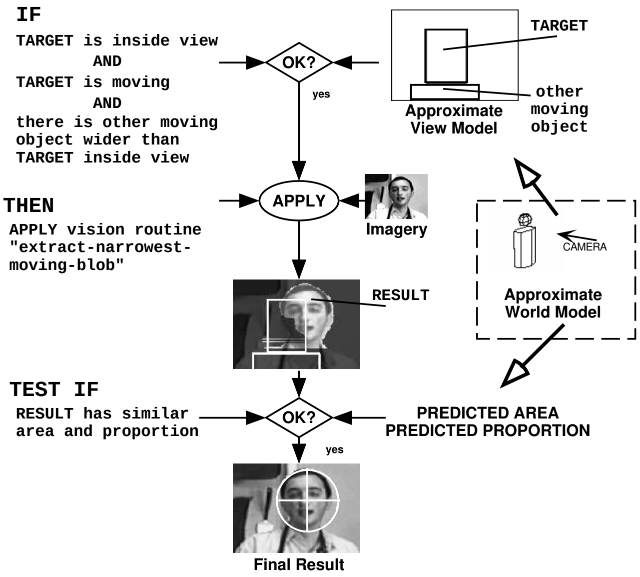

Figure 1: Example of an applicability rule for a vision routine and how the information from the approximate world model is used.

It is interesting to situate our scheme in the ongoing debate about reconstructionist vs. purposive vision discussed in [13] and in the replies in the same issue. Our proposal falls between the strictly reconstructionist and purely purposive strategies. We are arguing that reconstruction should exist at the approximate level to guide purposive vision routines: by building an approximate model of the scene vision systems can use task-specific, purposive vision routines which work reliably in some but not all situations.

3 Using Approximate World Models in Vision Systems

To formalize the control exercised by the approximate world models in a vision system, we define *applicability conditions* for each vision routine: the set of assumptions, that, if true in the current situation, warrants faith in the correctness of the results of that routine.

The idea is to have each vision routine encapsulated in an *applicability rule*, which describes pre-conditions (`IF` portion of the rule), application constraints (`THEN` portion), and post-conditions (`TEST IF`), in terms of general properties about the targeted object, other objects in the scene, the camera's point of view, and the result of the vision routine.

As an example, fig. 1 depicts the applicability rule of a vision routine, "`extract-narrowest-movingblob`", and how the rule is applied in a given situation. The routine "`extract-narrowest-moving-blob`" is designed to detect moving regions in a sequence of two consecutive frames using simple frame differencing, and then to divide the result into two areas, of which the narrowest is returned.

To use such a rule, the vision system consults the approximate model of `TARGET` (in the example case, the head of a person) to obtain an estimation of its projection into the image plane of the camera, and also to confirm if `TARGET` is moving. Moreover, the system also looks for other moving objects close to `TARGET`, constructing an *approximate view model* of the camera's view.

If the conditions are satisfied by the approximate view model, the routine is applied on the imagery according to the instructions in the `THEN` portion of the applicability rule which may also include information about routine parameters. The `RESULT` is then checked in the `TEST IF` portion of the rule, reducing the possibility that an incomplete specification of pre-conditions generates incorrect results. For instance, in the case of "`extract-narrowest-moving-blob`", often the lack of actual object movement makes the routine return tiny, incorrectly positioned regions which are filtered out by the post-conditions.

It is important to differentiate the concept of applicability rules from rule-based or expert-system approaches to computer vision [4, 15]. Although we use the same keywords (`IF`, `THEN`), the implied control structure has no resemblance to a traditional rule-based system: there is no inference or chaining of results. More examples of applicability rules can be found in [2].

4 A Working Example: SmartCams

Our approach of using approximate world models is being developed in a system we are constructing for the control of TV cameras. The ultimate objective is to develop a camera for TV that can operate without the cameraman, changing its attitude, zoom, and position to provide specific images upon human request. We call these robot-like cameras *SmartCams*. A "cooking show" is the first domain in which we are experimenting with our SmartCams.

The basic architecture of a SmartCam is shown in fig. 2. Considering the requirements of this application, the approximate world model represents the subjects and objects in the scene as 3-D blocks, cylinders, and ellipsoids, and uses symbolic frame-based representations (slots and keywords). The symbolic description of an object includes information about to which class of objects the object belongs, its potential use, and its roles in the current actions.

The 3-D representations are positioned in a 3-D virtual space corresponding to the TV studio. The cameras' calibration parameters area also approximately known. The precision can be quite low, and in our system the position of an object might be off by an amount comparable to its size.

For example, if there is a bowl present in the studio, its approximate world model is composed by a 3-D geometric model of the bowl and by a frame-like symbolic description. The 3-D geometric model approximates the shape of the bowl, and is positioned in the virtual space according to the available information. The objects in the approximate world model belong to different categories. For example, a bowl is a member of the "`handleable objects`" category. As so, its frame includes slots which describe whether the bowl is being handled by a human, and, if so, there is a slot which explicitly points to him/her.

The system which produced the results shown in this paper does not use real moving cameras, but simulates them using a moving window on wide-angle images of the set. Several performances of a 5-minute scene as viewed by three wide-angle cameras were recorded and digitized. The SmartCam output image is generated by extracting a

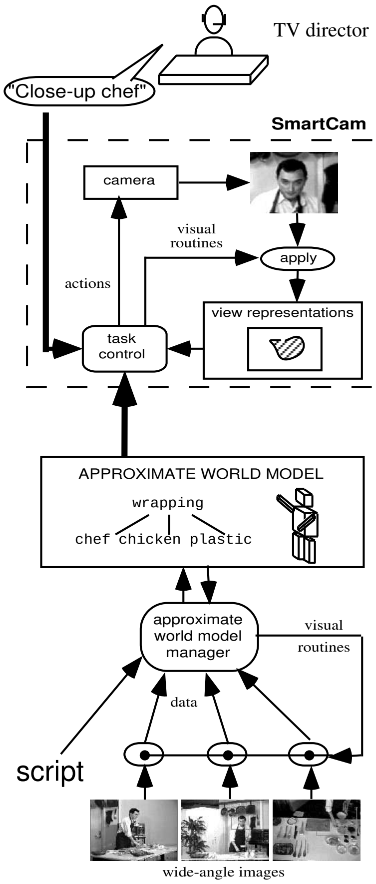

Figure 2: The architecture of a SmartCam. The bottom part of the figure shows the structure of the modules responsible for maintaining the approximate world models.

rectangular window of some size from the wide-angle images.

In fig. 3 we can see a typical result in the SmartCam domain where the inaccuracy of the approximate world models does not affect the final results obtained by the vision routines. Two SmartCams, `side` and `center`, were tasked to provide a close-up of the chef. Although the geometric model corresponding to the chef is quite misaligned, as can be seen by its projection into the wide-angle images of the scene (left side), both SmartCams, using routines similar to "`extract-narrowest-moving-blob`", produce correct results (the highlighted areas on the right of fig.3). Other examples of applicability rules and results can be found in [2].

5 Building and Maintaining Approximate World Models

Having shown how the information contained in approximate world models can be exploited by a vision system performing tasks in a dynamic environment, a fundamental issue remains: how does one construct an initial model and then maintain such a model as time progresses and the scene changes.

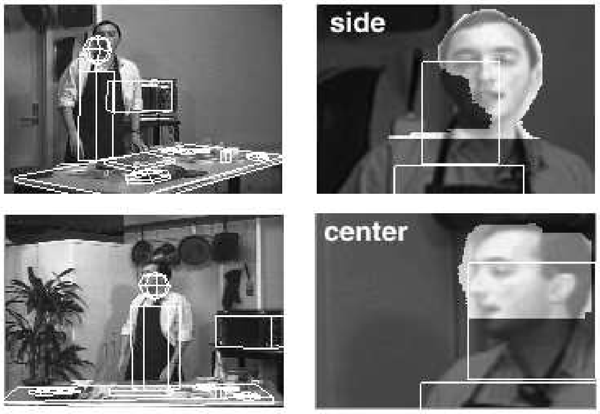

Figure 3: Example of response to the call "`close-up chef`" by two different cameras, `side` and `center`. The left images show the projection of the approximate models on the wide-angle images. The right images display the result of vision routines as highlighted regions, compared to the predicted position according to the approximate model of the head and the trunk, shown as rectangles.

Contextual and semantic information is rarely employed in model construction because of its inability to provide accurate geometric data. If the geometric accuracy requirements are relaxed, as it is in the case of approximate world models, semantic information can be used to predict possible positions and attitudes of objects.

Furthermore, we view one of the roles of contextual knowledge to be that of providing the basic relationships between objects in a given configuration of the world. For example, while pounding meat the chef remains behind the table, positioned near the cutting board, and the meat mallet is manipulated by the hands. As long as such a context remains in force, these relationships hold, and the job of maintaining the approximate world model reduces to, predominantly, a problem of tracking incremental changes within a known situation.

Occasionally, though, there are major changes in the structure of the world which require a substantial update in the structure of the approximate models. For example, a new activity may have begun, altering most expectations, including, for example, the possible location of objects, which objects are possibly moving, and which objects are likely to be co-located making visual separation unlikely (and, more interestingly, unnecessary). In our view, the intervals between *major contextual changes correspond to the fundamental actions* performed by the subjects in the scene; the contextual shifts themselves reflect the boundaries between actions.

Thus, maintaining approximate world models requires two different methods of updating: one related to the tracking of incremental changes within a fixed context, and the other responsible for performing the substantial changes in the context required at the boundaries between different actions. Of course, it is also necessary to have methods to obtain the initial approximate model. The three required mechanisms are briefly discussed below.

5.1 Initializing Approximate World Models

Whenever a vision system is designed using approximate models, it is necessary to face the issue of how the models are initialized when the system is turned on. Current computer vision methods can be employed in the initialization process, if the system is allowed a reasonable amount of time to employ powerful vision algorithms without contextual information.

In our current version of the SmartCams the 3-D models of the subjects and objects are determined and positioned manually in the first frame of the scene. All changes to the model after the first frame are accomplished automatically using vision and processing the linguistic information as described later in this paper.

5.2 Tracking Incremental Changes

As proposed, while a particular action is taking place it is presumed that the basic relationships among the subjects and objects do not change. During such periods most of the updating of the approximate models can be accomplished by methods able to detect incremental changes, such as visual tracking algorithms. Though simple, those small updates are vital to maintain the approximate model in an useful state, since approximate position information is normally an essential part of the applicability conditions of the task-specific vision routines.

In the SmartCam domain, the update of the incremental changes in the approximate world model, and especially in its 3-D representations, is accomplished by vision tracking routines able to detect movements of the main components of the scene, as shown in the bottom part of the diagram in fig. 2. The two-dimensional motions of an object detected by each of the wide-angle cameras are integrated to determine the movement of the object in the 3-D world. More details can be found in [2].

Note that the use of an approximate world model may require additional sensing and computation which might not be performed to directly address current perceptual tasks: e.g. the position of the body of the chef is maintained even though the current task may only involve framing the hands. We believe that this additional cost is compensated by the increase in the competence of the vision routines.

5.3 Actions and Structural Changes

Tracking algorithms are likely to fail whenever there is a drastic change in the structure of the scene, as, for example when a subject leaves the scene. If this situation is recognized, the tracking mechanism of that subject should be turned off avoiding false alarms.

It is not the objective of this paper to argue about the different possible meanings of the term *action*. Here, actions refer to major segments of time which people usually describe by single action verbs as discussed by [8, 14, 5].

A change in the action normally alters substantially the relationships among subjects and objects. For example, we have a situation in the cooking show domain where the chef first talks to the camera, and then he picks up a bowl and starts mixing ingredients. While the action "talking" is happening, there is no need for the system to maintain explicit 3-D models for the arms and the hands of the chef: the important elements involved are the position and direction of the head and body.

When the chef starts mixing ingredients, it is essential that the approximate world model includes his hands, the mixing bowl, and the ingredients. Fortunately, "mixing" also sets up clear expectations about the positions of the hands in relation to the trunk of the chef, to the mixing bowl, to the ingredients' containers, and to table where they are initially on. As this example illustrates, known contextual changes actually improve the robustness of the system since with each such event the system becomes "grounded": there is evidence independent of, and often more reliable than, the visual tracking data about the state of the scene.

6 Linguistic Sources of Action Information

Without constraints from the domain, determining the actions which might occur can be difficult. However, in many situations the set of actions is severely restricted by the environment or by the task, as, for example, in the case of recognizing vehicle maneuvers in a gas-station as described in [7].

The SmartCam domain exemplifies another type of situation, one in which there is available a linguistic description of the sequence of actions to occur. In a TV studio, the set of occurring actions — and, even, the order of the actions — is determined by the script of the show. We find the idea of having vision systems capable of incorporating linguistic descriptions of actions very attractive: linguistic descriptions of actions are the most natural to be generated by human beings. In particular, this form of is suitable in semi-automated vision-based systems.

The use of linguistic descriptions requires their automatic translation into the system's internal representational of actions. This is mostly a Natural Language Processing issue, although the feasibility is certainly dependent on the final representation used by the vision system. In the SmartCam domain the final objective is to employ TV scripts in the format they are normally written (see figure 4).

6.1 Representing Actions

Many formalisms have been developed to represent action, some targeting linguistic concerns [10], computer graphics synthesis [5], or computer vision recognition [11]. Currently we employ a simple representation based on Schank's *conceptualizations* as described in [10]. In spite of its weaknesses — see [16] — Schank's representation scheme is interesting for us because the reduced number of primitive actions helps the design of both the translation and the inference procedures.

Our representation for actions uses *action frames*, a frame-based representation where each action is represented by a frame whose header is one of Schank's primitive actions — `PROPEL`, `MOVE`, `INGEST`, `GRASP`, `EXPEL`, `PTRANS`, `ATRANS`, `SPEAK`, `ATTEND`, `MTRANS`, `MBUILD` — plus the attribute indexes `HAVE` and `CHANGE`, and an undetermined action `DO`.

Figure 5 contains two examples of action frames. The figure contains the representation for two actions of the script shown in fig. 4, `''chef wraps chicken with a`

Cooking-show scenario with a table on which there are bowls, ingredients, and different kitchen utensils. A microwave oven is in the back. Cam1 is a centered camera, cam2 is a left-sided camera, cam3 is a camera mounted in the ceiling. Chef is behind the table, facing cam1.

...

Chef turns back to cam1 and mixes bread-crumbs, parsley, paprika, and basil in a bowl.
"Stir together 3 tablespoons of fine dry bread crumbs, 2 teaspoons snipped parsley, 1/4 teaspoon paprika, and 1/8 teaspoon dried basil, crushed. Set aside."

Chef wraps chicken with a plastic bag.
"Place one piece of chicken, boned side up, between two pieces of clear plastic wrap."

Chef puts the chicken on the chopping-board and shows how to pound the chicken.
"Working from the center to the edges, pound lightly with a meat mallet, forming a rectangle with this thickness. Be gentle, meat become as soft as you treat it."

Chef pounds the chicken with a meat-mallet.

...

Figure 4: The script of a TV cooking show.

```
;; "chef wraps chicken with a plastic bag"
(do
 (actor chef)
 (result
  (change (object chicken)
   (to (and
        (contained plastic-bag)
        (physical-contact plastic-bag))))))

;; "chef pounds the chicken with a meat-mallet"
(propel
 (actor chef)
 (object meat-mallet)
 (from (location ?not-in-contact))
 (to (location ?-in-contact))
 (result
  (change
   (object chicken)
   (from (flatness ?X))
   (to (flatness (greater ?X)))))
 (instrument
  (have (object ?not-in-contact)
   (attribute
        (negation (physical-contact chicken)))))
 (instrument
  (have (object ?in-contact)
   (attribute (physical-contact chicken))))
 (instrument
  (have (object chicken)
   (attribute
        (physical-contact chopping-board)))))
```

Figure 5: Action frames corresponding to two actions from the script shown in fig. 4.

`plastic bag''` and ``` ``chef pounds the chicken with a meat-mallet'' ```. Each action frame begins with a primitive action and contains different slots which supply the essential elements of the action. Undetermined symbols begin with a question mark (?); those symbols are defined only by the relationships they have with other objects. The `actor` slot determines the performer of the action while the `object` slot contains the object of an action or the owner of an attribute. The action frames resulting from an action are specified in the `result` slot, and action frames which are part of the definition of another action frame are contained in `instrument` slots.

In the first example, "wrapping" is translated as some action (unspecified by `DO`) whose result is to make chicken be both contained in and in physical contact with a plastic bag. In the second example, "pounding" is represented as an action where the chef propels a meat mallet from a place which is not in contact with the chicken to a place which is in contact, and whose result is an increase in the flatness of the chicken.

The current version of our SmartCams translates a simplified version of the TV script of fig. 4 into the action frames of fig. 5 using a domain-specific, very simple parser. Building a translator able to handle more generic scripts seems to be clearly a NLP problem, and, as such, it is not a fundamental point in our research.

Part of our current research is focused on designing a better representation for actions than the action frames described in this paper. We are still debating the convenience of using Schank's primitives to describe every action. Also, action frames need to be augmented by incorporating at least visual elements, as in [5], and time references, possibly using Allen's interval algebra, [1].

6.2 Using Action Frames Obtained From a Script

From the examples shown above, it is clear that linguistic descriptions of actions obtained from scripts do not normally include detailed information about the position, attitude, and movement of the persons and objects in the scene. Linguistic accounts of actions normally describe only the essential changes in the scene, but not the implications of those changes.

Therefore, to use information from scripts it is necessary to have an inference mechanism capable of extracting the needed details from the action frames. In particular, to use the action frames generated from the TV script in the SmartCam domain, it was necessary to implement a simple inference system. It is important to make clear that our inference system is extremely simple and designed only to meet the demands of our particular domain. The system was designed to infer position and movement information about human beings' hands, and physical contact and proximity among objects.

The inference system is based on Rieger's inference system for Schank's conceptualizations, [9]. The inferred action frames are sub-actions, or `instrument` actions of the actions from which they are produced. To guarantee termination in a fast time, the inference rules are applied in a pre-determined sequence, in a 1-pass algorithm.

As a typical case, the system uses as its input the action frame corresponding to the sentence ``` ``chef wraps chicken with a plastic bag'' ``` (as shown in fig. 5) and deduces that the chef's hands are close to the chicken. The appendix depicts a more complex example where from the action of pounding the system obtains the fact that the hands are close to the chopping board. From the PROPEL action frame shown in fig. 5, the inference system deduces some contact relations between some objects, which imply physical proximity.

The SmartCam's inference system is certainly very simple and works only for some scripts. However, the ability of approximate models to handle inaccurate information helps the system to avoid becoming useless in the case of wrong inferences. For instance, in the "pounding" example, only one of the hands is in fact close to the chopping board: the hand which is grasping the meat mallet is about 1 foot from the board. But, as we have seen above, such errors in positioning are admissible in the approximate world model framework.

6.3 Determining the Onset of Actions

In the examples above, the information from the script was represented and augmented by simple inference procedures. However, to use script information it is necessary to "align" the action frames with the on-going action, that is, the vision-based system need to recognize which action is happening in any given moment in time.

For the work presented here we have relied on manual alignment of the action frames to the events in the scene. All the results shown in this paper use a *timed script*, an extension of the script of fig. 4 which includes information about when each of the action is happening. This is simplified by the fact that the we are using the simulated version of the SmartCams where the visual data is pre-recorded, enabling manual annotation.

The problem of visual recognition of actions is also a current object of our research. The alignment problem mentioned above can be viewed as a sub-problem of the general action recognition problem where the order of the actions is known in advance.

7 SmartCams in Action

The current version of our SmartCams handles three types of framing (close-ups, medium close shots, medium shots) for a scenario consisting of the chef and about ten objects. All the results obtained so far employ only very simple vision routines similar to "`extract-narrowest-moving-blob`", based on movement detection by frame differencing.

Figure 6 shows typical framing results obtained by the system. The leftmost column of fig. 6 displays some frames generated in response to the call "`close-up chef`". The center column of fig. 6 contains another sequence of frames, showing the images provided by the SmartCam tasked to provide "`close-up hands`".

The rightmost column of fig. 6 is the response to a call for a "`close-up hands`". In this situation, the action `''chef pounds the chicken with a meat-mallet''` is happening. As shown above, this action determines that the hands must be close to the chopping board. This information is used by the system to initialize expectations for the hands in the beginning of the action (both in terms of position and movement), enabling the tracking system to detect the hands' position based solely in movement information.

Figure 6: Responses to the calls "`close-up chef`", "`close-up hands`", and "`close-up hands`". Refer to background objects to verify the amount of correction needed to answer those calls appropriately. The grey areas to the right of the last frames of the first "`close-up hands`" sequence correspond to areas outside of the field of view of the wide-angle image sequence used by the simulator.

One 80-second long video sequence is shown in the videotape distributed with these proceedings. It is also available on the WWW-web at:
`http://www-white.media.mit.edu/`
`vismod/demos/smartcams/smartcams.html`
The web-site also contains another performance of the same script where the chef is wearing glasses, and the actions are performed in a faster speed. The sequences were obtained by requesting the SmartCams to perform specific shots (displayed as subtitles); the cuts between cameras were selected manually. Both sequences clearly show that acceptable results can be obtained by our SmartCams in spite of the simplicity of the vision routines employed.

8 Conclusion

Approximate world models made the development of SmartCams feasible. Using the information about actions from the script of the show and the control information in the approximate world model, it has been possible to employ simple, fast — sometimes unreliable — vision routines to obtain the information required by TV framing.

One of the major accomplishments of our research is the end-to-end implementation of a system able to deal with multiple levels of information and processing. A SmartCam is able to use contextual information about the world from the text of a TV script, and to represent the information in a suitable format (approximate world models); updating the world model is accomplished through visual tracking, and the approximate world models are used in the selection and control of vision routines, whose outputs control the movement of a simulated robotic camera. The system processes real image sequences with a considerable degree of complexity, runs only one order of magnitude slower than real time, and produces an output of good quality in terms of TV standards.

Appendix

A Inferences in the "Pounding" Example

The following is a manually commented printout of the action frames generated by the SmartCam's inference system, using as input the action frame corresponding to the sentence ``chef pounds the chicken with a meat-mallet''. Only the relevant inferences are shown from about 80 action frames actually generated. The transitive rule used for the inference of proximity is sensitive to the size of the objects, avoiding its use if one of the objects is larger than the others.

0 : action frame obtained from the script
```
(propel
 (actor chef)
 (object meat-mallet)
 (to (location ?in-contact))
 (from (location ?not-in-contact))
 (result
  (change (object chicken)
   (from (flatness ?X))
   (to (flatness (greater ?X)))))
 (instrument
  (have
   (object ?in-contact)
   (attribute (phys-cont chicken))))
 (instrument
  (have
   (object ?not-in-contact)
   (attribute (negation (phys-cont chicken)))))
 (instrument
  (have
   (object chicken)
   (attribute (phys-cont chopping-board)))))
```
1 : propelling an object (0) requires grasping
```
(grasp
 (actor chef)
 (object meat-mallet)
 (to hands))
```
2 : grasping (1) requires physical-contact
```
(have
 (object hands)
 (attribute (phys-cont meat-mallet)))
```
3 : physical-contact (0) implies proximity
```
(have
 (object ?in-contact)
 (attribute (proximity chicken)))
```
4 : physical-contact (0) implies proximity
```
(have
 (object chicken)
 (attribute (proximity chopping-board)))
```
5 : physical-contact (2) implies proximity
```
(have
 (object hands)
 (attribute (proximity meat-mallet)))
```
6 : the end of propelling (0) implies proximity
```
(have
 (object meat-mallet)
 (attribute (proximity ?in-contact)))
```
7 : transitiveness of proximity, (3) and (6)
```
(have
 (object chicken)
 (attribute (proximity meat-mallet)))
```
8 : transitiveness of proximity, (4) and (7)
```
(have
 (object chopping-board)
 (attribute (proximity meat-mallet)))
```
9 : transitiveness of proximity, (5) and (8)
```
(have
 (object chopping-board)
 (attribute (proximity hands)))
```

References

[1] James F. Allen. Towards a general theory of action an time. *Artificial Intelligence*, 23:123–154, 1984.

[2] Aaron Bobick and Claudio Pinhanez. Using approximate models as source of contextual information for vision processing. In *Proc. of the ICCV'95 Workshop on Context-Based Vision*, pages 13–21, Cambridge, Massachusetts, July 1995.

[3] Aaron F. Bobick and Robert C. Bolles. The representation space paradigm of concurrent evolving object descriptions. *IEEE PAMI*, 14(2):146–156, January 1992.

[4] B. A. Draper, R. T. Collins, J. Brolio, J. Griffith, A. R. Hanson, and E. M. Riseman. Tools and experiments in the knowledge-directed interpretation of road scenes. In *Proc. of the DARPA Image Understanding Work-*

shop, pages 178–193, Los Angeles, California, February 1987.

[5] Jugal Kumar Kalita. *Natural Language Control of Animation of Task Performance in a Physical Domain.* PhD thesis, University of Pennsylvania, Philadelphia, Pennsylvania, June 1991.

[6] David Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. In *Proc. R. Soc. Lond. B*, volume 200, pages 269–294, 1978.

[7] Hans-Hellmut Nagel. A vision of 'vision and language' comprises action: An example from road traffic. *Artificial Intelligence Review*, 8:189–214, 1994–5.

[8] Darren Newtson, Gretchen Engquist, and Joyce Bois. The objective basis of behavior units. *Journal of Personality and Social Psychology*, 35(12):847–862, December 1977.

[9] Charles J. Rieger III. Conceptual memory and inference. In *Conceptual Information Processing*, chapter 5, pages 157–288. North-Holland, 1975.

[10] Roger C. Schank. Conceptual dependency theory. In *Conceptual Information Processing*, chapter 3, pages 22–82. North-Holland, 1975.

[11] Jeffrey Mark Siskind. Grounding language in perception. *Artificial Intelligence Review*, 8:371–391, 1994–5.

[12] Thomas M. Strat and Martin A. Fischler. Context-based vision: Recognizing objects using information from both 2-d and 3-d imagery. *IEEE PAMI*, 13(10):1050–1065, October 1991.

[13] Michael J. Tarr and Michael J. Black. A computational and evolutionary perspective of the role of representation in vision. *CVGIP: Image Understanding*, 60(1):65–73, July 1994.

[14] Robert Thibadeau. Artificial perception of actions. *Cognitive Science*, 10:117–149, 1986.

[15] J. K. Tsotsos. Knowledge organization and its role in representation and interpretation of time-varying data: The ALVEN system. *Computational Intelligence*, 1:16–32, 1985.

[16] W. Wilks. A preferential, pattern-seeking semantics for natural language inference. *Artificial Intelligence*, 6(1):53–74, 1975.