Presenting generalized human activities in virtual environment

Vladimír Štěpán[*]
Czech Technical University in Prague,
Czech Republic

Jiří Žára[†]
Czech Technical University in Prague,
Czech Republic

Václav Hlaváč[‡]
Czech Technical University in Prague,
Czech Republic

Abstract

The EU cognitive vision project ActIPret (aimed at interpreting and understanding human manipulation activities) included a module for presentation of results in virtual reality (VR). This paper is mainly a detailed description of our solution - the ActIPret VR presentation module. Moreover we explain the specifics of cooperation between two areas with totally different view of the subject, such as cognitive vision and virtual reality. Thus the paper can be viewed as an overview of our experience with this complicated sort of task.

Keywords: Cognitive vision, activity interpretation, virtual reality, human animation, inverse kinematics

1 Introduction

This paper presents the virtual reality (VR) presentation module of the ActIPret system.

The ActIPret (Interpreting and Understanding Activities of Expert Operators for Teaching and Education) [Act] was 5^{th} framework project in cognitive vision call run 2001 - 2004. The system's ability to learn should be demonstrated in following simplified setting. The ActIPret system should perceive activity demonstrated by an expert and represent it in the form of the activity plan (AP). When the trainee attempts to perform the activity, the system should perceive it and, by comparison with the stored expert activity, provide hints.

Learning and perception is tested in ActIPret scenarios which encapsulate the activity. The *'insert CD scenario'* is the simplest one used in the project. The scenario comprises the activity of inserting CD into a player: human operator opens the CD player, selects a CD, inserts it in the player and closes the player.

The goal of the project was basically to create generalized activity plan describing the semantics of observed activity. VR module was originally supposed to visualize the outputs of ActIPret system, which would be highly abstract AP. There definitely is a gap between such requirements and possibilities of VR presentation. The first problem to solve was to determine the level of abstraction of actual VR presentation.

[*]e-mail: stepanv@fel.cvut.cz
[†]e-mail: zara@fel.cvut.cz
[‡]e-mail: hlavac@fel.cvut.cz

The ActIPret framework focuses on the phenomena relevant to the activity. This includes recognizing and localizing certain objects, tracking operator's hands, detecting interaction events, etc. Good, realistic and understandable VR presentation needs more information though. Thus the task of our VR presentation module could be viewed as activity reconstruction with incomplete input data.

The document is structured as follows: The Section 2 briefly summarize the task and the design structure of ActIPret VR presentation module. In the Section 3, we discuss the problem of visualization of generalized data, such as ActIPret output. Next, in Section 4, we mention the possible solutions to our particular task. The main part of this paper is the Section 5, which brings the description of the solution actually implemented as ActIPret VR presentation module.

2 VR presentation module

The VR module is not directly coupled to other modules of the ActIPret framework. It can be considered a stand-alone program for presentation the project results and generally for arbitrary human-oriented activities. It was decided to build the VR module on VRML standard.

It was agreed that the presentation of the human activity in VR will be in a form of reconstruction and replay of the activity. The presentation consists of three logical components: *3D geometry, animation* and *interactions*. The 3D model of the scene can be captured off-line using computer vision techniques. The model includes 3D geometry and appearance of the scene and involved objects. Example objects in the 'insert CD scenario' are the CDs, CD player, etc.

The animation part includes all the motion in the scene. During the activity all the objects move as a result of human action. Therefore we need to animate the virtual human only. Other objects can be moved between locations in quite realistic fashion by attaching them to the moving virtual human geometry. This belongs to the third part of activity description - the interactions.

3 Visualization Problems

The task of the VR presentation module was "to visualize the project output in VR". The word *visualization* usually means the graphical explanation of information. The project's output is the activity plan. The activity plan describes the semantics of the activity. It is abstract and basically comprises of the sequence of events. Example of the 'insert CD scenario' activity plan:

```
Hand pressed button openButton
Hand picked up object CD Hand put
down object CD on object Player
Hand pressed button openButton
```

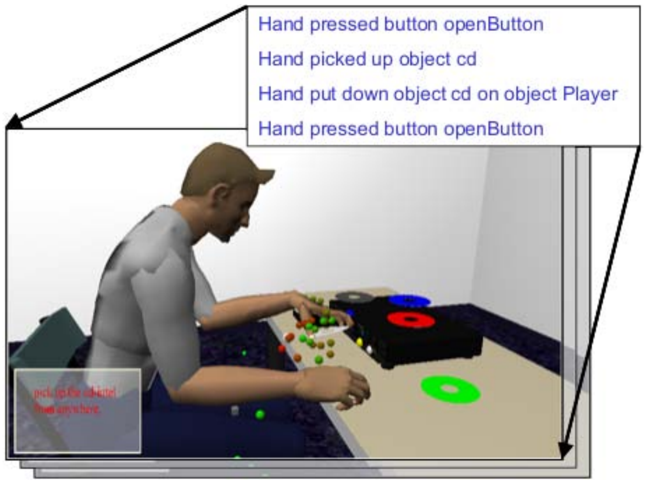
Hand pressed button openButton
Hand picked up object cd
Hand put down object cd on object Player
Hand pressed button openButton

Figure 1: The activity plan visualization. The figure shows the main idea of ActIPret VR module task. Abstract textual activity plan (in the box above) is be transformed to the animated scene indicated here as a sequence of images.

Obviously this does not provide enough information for *visualizing* the activity in VR as can be clearly demonstrated by Fig. 1. We can assume there were some objects in the scene and some interactions among these objects. Let us assume we are an educated (we have considerable amount of knowledge) human, who knows what is it to press button or pick up something and knows what impact these events have on the objects involved. The activity plan can give us pretty good idea what is going on in the scenario.

While the activity plan alone can give us an idea of the activity, still it does not allow us to visualize the scenario. The purpose of every visualization is to help our mind in instantiation, which mostly means decreasing the level of abstraction. If we imagine (visualize in our mind) the activity then we create imaginary objects, imaginary operator and we let the object move or be moved between imaginary locations. The locations of these objects are chosen with respect to our experience with handling real objects, for instance all is usually within operator's comfortable reach. We are using a lot of knowledge in this visualization process.

The VR presentation module needs this knowledge too, but on the contrary to human mind it does not have it. The activity plan is too abstract to be visualized reasonably clearly. We need to backtrack the way to the level of abstraction of an activity plan. The activity represented by the activity plan needs to be instantiated somehow.

3.1 The Scene

There are objects referenced in the activity plan. The instantiation of activity should include localization of these objects. The knowledge of their initial placement allows us to create the scene. This scene would be rather less realistic. Only the object listed in an activity plan and therefore relevant to the activity would be present. There appear to be objects of two categories in the scene. There are objects relevant to the activity and therefore detected by the system. The other category would cover all the objects that are present, but are not subject of system's attention.

This can result into a conflict with the explanatory purpose of visualization. People are used to some form of incidence among the objects in surrounding world, that would most likely be missing in such scene. For explanation the CD should not hang in midair, it should be supported by the table instead. Thus the presence of the other kind of objects in the scene might be necessary for the sufficient level of realism.

We should also mention the fact, that the activity plan gives us absolutely no information on the actual appearance of the objects. To visualize an activity plan it would be necessary to use the set of predefined models of all objects playing functional or visual role in the scenario. At this point we create the group of *scenario specific data*.

3.2 Animation

After the scene is ready we can make it move. This is the task of animating the virtual human, as all the motion of objects is directly derived from motion od human.

The activity plan provides no information usable for animating the virtual human. In the previous section we reduced the level of abstraction of the activity plan with known initial object locations. Can this information help us animate the scene? We can presume that if the hand interacts with an object, as suggested by activity plan, it is located somewhere near the object. The expression "somewhere near" is just as vague as the actual information on the location the activity plan gives us. What exactly is this "somewhere near" depends on the type of object and the type of event.

We have to know more about the activity, at least the position of the hand in the moment when an event from the activity plan occurred. For more realism and avoidance of possible collisions with objects, knowing the trajectory of the hand during the activity would be useful. Large portion of the animation task, particularly the finger animation, can be performed by connecting gestures associated with the detected events. Since these gestures can be predefined, they belong to the same category as the object models and can be viewed as a scenario specific information.

3.3 Interaction

In the real scene, the activity is based on interaction of human operator and the objects in the environment. By implementing these interactions in VR we can easily animate the scene just by animating the virtual human. The interactions are the semantics of the scene and as such they are well described by the activity plan.

4 Available Options

The VR presentation module has a special position as it is not an integral part of the project's framework. Therefore it is quite limited in its possibilities. When designing it, we decided to avoid having a special requirements for input data. The goal was to use the project output and, if necessary some of the data used by another part of the framework available in some form for use outside of framework. Definitely the VR module has to be scenario independent. As an addition to this, the generality of the input should be as close as possible to the generality of activity plan.

4.1 Sources of Data

The sources of data for VR presentation of the activity are very limited within the framework. We have already mentioned the *activity* *plan* and demonstrated it on the example (see Section 3). It describes the semantics of the activity quite well and thus it provides good description of the interactions between human operator and the objects.

One of the lower level modules in the framework, the *Object Relation Generator* (ORG), generates the *log*, that basically sums up the results of object detection and recognition and tracking operator's hand. The list of objects along with their positions can be found here for the scene reconstruction task. Also the trajectory of the hand can be obtained here. We can use it to animate a virtual human with an inverse kinematics (IK) algorithm.

Neither the activity plan nor ORG log gives us enough information for acceptably realistic VR presentation. We introduce third source of information that we call *scene definition file* (SDF). It is the text file that describes the situation in the time when observed activity starts. The SDF is a complete list of all objects in the scene with all the information on them that will be used to accomplish the task of activity reconstruction. This includes relationships between objects and initial positions of objects not detected and listed in ORG log.

As it is suggested in the text above, the VR presentation depends on number of predefined structures. This includes most of the scene with virtual human and some building blocks for hand animation (the gestures).

4.2 Possible Approaches

There are two possible approaches to VR presentation module, 'general' and 'instance-based'. Obviously they differ in level of abstraction. In practice the difference is in the data used. The 'instance-based' approach works with observed data from the ORG log, most importantly the hand trajectory. On the contrary, 'general' approach should abstract from using the observed data and therefore generates its own hand trajectory.

The **'instance-based'** approach is basically the reconstruction of observed activity in VR. We use detected object positions to place the predefined models into the scene. The models of objects are a little more than just geometry, as some of them implement some functionality, for example CD player model for the 'CD scenario' can open and close the tray at the proper event.

The virtual human is animated using the observed hand trajectory. This trajectory has a positive effect as it brings the hand to the appropriate position for interaction with the object. Moreover it reduces the risk of collisions with other objects in the scene. This risk is not reduced to zero, because only general models that might not always match the actual models in shape, size and proportion are used. The precise scene reconstruction is not the goal of the module, so the models can also be placed with significant error.

The negative effect of using hand trajectory is the need for a time information of the occurrence of interaction events. This information is present in the framework, but does not pass to any output file because the cognitive science experts claim it is not needed for expected generalization. The need for this information is in conflict with our rule of avoiding special requirements. Besides that the system knows the time when the event was detected, which is delayed after the hand trajectory. For the needs of VR presentation, the time of the event occurrence would be more desirable.

Animating the virtual human with the hand trajectory is a typical task for an IK algorithm. Although the system tracks the fingertips as additional clues for the cognitive tasks, the fingertips are not recognized and labelled, their trajectories are incomplete, therefore it is not possible to use these trajectories to animate hand with IK as it was done with the rest of the body. We combine the IK animation with the predefined gestures associated with events listed in the activity plan. These gestures are inserted to the animation at the key-frames indicated by the time of event occurrence.

The **'general'** approach does not use the observed hand trajectory. It starts with the events listed in the activity plan to generate its own trajectory instead. This way we don't need the time information, because new animation creates its own timing.

New problems appear though. We do not know the position of hand relative to the object at the moment of interaction. This position and the posture of the hand depends on the object and event. Therefore we have put together the models of objects with the description of hand postures and positions according to the concept of *smart objects* [Goncalves et al. 2001]. Thus the model of object encapsulates not only the appearance description, but also the hints for virtual human for every possible type of interaction.

The activity plan is the sequence of events. Each event marks the moment when the hand was near some object performing some action. The object and action are indicated in activity plan. With the objects carrying the information about hand position and posture, the event can be viewed as a key-frame of the new animation. The posture of the body at the key-frame is then computed by IK algorithm and combined with the hand posture. The input for IK and the hand posture are obtained from the object.

Queueing the events as key-frames would not alone yield very realistic output. Therefore each action has its own general animation model and the object specific frames are inserted to this model. The resulting animation is created by queueing these models.

We should conclude this consideration stating that we chose the 'instance based' approach for implementation. The choice was done mainly because it was less demanding in terms of object model complexity and animation timing. The trade off was the necessity to use less general (instance) data such as the hand trajectory.

5 Implementation Issues

The **'instance-based'** approach was used to implement the VR presentation module. Although it is not general enough and various problems may occur (collisions), it was possible to implement within our time schedule and the available data. Figure 2 shows basic scheme of implemented design.

The activity presentation module is divided into two parts. The first step is reconstruction of initial setup combining the detected information with the user input. The user specifies what other object are in the scene, their relationships to the detected objects and their exact positions. Moreover the user must define the initial pose and position of virtual humanoid and the kinematic chains to be animated. This first step creates SDF. Next the reconstructed initial setup is animated and the result can be viewed and saved in VRML based format.

The ActIPret VR presentation module as well as the visual SDF editor is Java application connected via External Authoring Interface (EAI) to VRML browser. The browser that has been used is Parallel Graphics Cortona VRML Client (version 4) that also provides the EAI classes. Due to the COM architecture of the Cortona EAI classes, the application is not platform independent and runs on Windows system only.

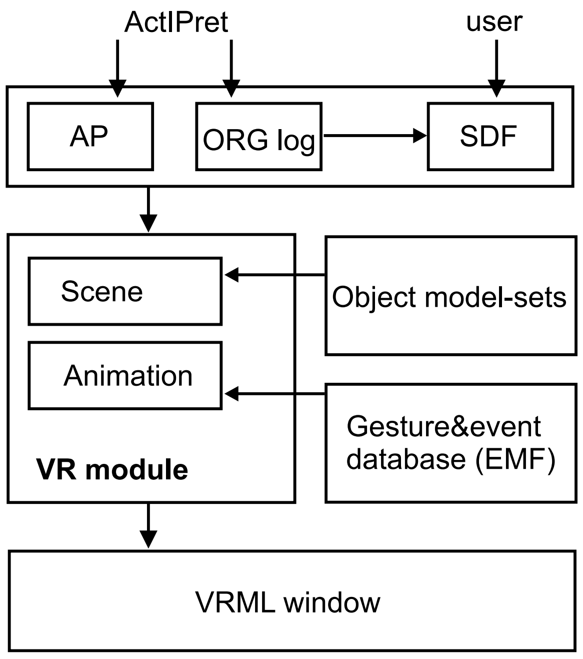

Figure 2: Diagram of the VR module indicating the relationships of input blocks. (EMF - event model file, see 5.3.2)

5.1 Input Data

The VR module uses three input files to create the VR presentation of observed activity - the *activity plan (AP)*, the *ORG log* and *Scene Definition File (SDF)*.

As the VR module presents instances of observed activities the three input files are stored in one directory belonging to this instance. The AP and ORG log are generated by framework and thus can be considered instance specific. On the other hand, SDF could be one for all instances. We added it to the other two files to allow the user to make changes to make the presentation instance specific too (use of various model sets, etc.).

The AP should be abstract and therefore scenario independent, but VR module considers it at certain aspects as instance specific. The main reason for this is the demonstration purpose of VR module. If the system happens to produce incomplete AP, when analyzing an observation, the VR module should be able to show the error.

5.1.1 Activity Plan

Activity plan describes the sequence of events. The file contains two versions - ‘conceptual language version’ and ‘natural language version’. The VR module uses the first to understand the events and create their VRML version and the second to be displayed in text window during the presentation. Example of an Activity Plan for

CD scenario would be:

```
====================================================
Raw Concepts

Line 0: PRESSBUTTON, Hand 0, Object ejectButton
Line 1: PICKUP, Hand 1, Object CD, Location undef
Line 2: PUTDOWN, Hand 1, Object CD, Location tray
Line 3: PRESSBUTTON, Hand 0, Object ejectButton
====================================================
Natural Language Version

Using first hand, press the ejectButton.
Using second hand, pick up the CD.
Using second hand, put the CD down on tray.
Using first hand, press the ejectButton.
====================================================
```

The lines of the “Raw Concepts” are comma separated lists of items that describe the event. First item indicates the type of event, the others are the objects involved in the event (hand, manipulated object and location of event).

5.1.2 ORG Log

The ORG log is an output of Object Relation Generator (ORG) module. It consists of list of following entries.

```
objects: ComponentID#ObjectID#ModelID#
appearance event: start-time end-time [pose]
...
end_of_object
```

This log was not created for use by VR module, therefore VR module often uses the information from the log in its specific way. Each ORG log entry is identified by 3 IDs, that are related to various aspects of the system and if. For purposes of VR module these IDs (combined) serve as one unique identifier. Moreover the names of the VR models are closely related to the ModelID which is originally an ID of recognition model used for an object.

The IDs are followed by one or more ‘appearance events’ that indicate the position of the object over a time interval.

Example:

```
4#0#cdplayer#
0.000 20.000 [1.29 0.36 0.72]
end_of_object
```

This example says that framework component with ID 4 detected an object (and labelled it with ID 0) that corresponds to recognition model of CD player. This object was detected in a time interval from 0 to 20 seconds on the position 1.29 0.36 0.72. For the VR module it means that the model of CD player must be inserted in the scene to given position and will stay there (no other appearance detection after 20 seconds).

5.1.3 SDF

The Scene Definition File (SDF) is used solely by VR module. It was designed to contain all the information necessary to reconstruct the scene that is not included in other sources. It is created by user with the help of special editor that takes in account the knowledge the system already has (ORG log).

SDF lists all objects in the scene, partly taken from ORG log, partly selected by user. Besides that it has an entry specifying the set of VRML models that should be used to represent these objects in the scene. The object descriptions are encapsulated in tags: <**object**> </**object**>

Between these tags there are object attributes followed by their values. These attributes should provide necessary information that cannot be retrieved from the framework. This includes associating the ActIPret ModelIDs with the VRML model names, human parts with H-Anim end-effectors for inverse kinematics (IK) animation, as well as defining relations between objects (subparts) and specifying various model parameters.

List of object attributes (alphabetical order):

- **color** - the value is a color of object written in three float numbers (RGB). In case of objects with models with variable color.
- **end_effector** - the name of H-Anim Site node within the definition of humanoid - an end-effector for IK animation.
- **fixed** - no value. The attribute forf objects not detected by ActIPret system.
- **ID** - value of this tag is the string that stands for ModelID in ActIPret framework (appears in the ORG log).
- **ik_base** - the name of H-Anim Joint node within the definition of humanoid - the base of a kinematic chain for IK animation.
- **orientation** - the value is four float numbers - orientation of fixed objects within the scene.
- **part_of** - if the object is a functional part of another object, this attribute has the same value as ID of its superordinate object.
- **pose** - the value is the filename of VRML file with description of a starting posture of H-Anim humanoid
- **position** - the value is three float numbers - position of fixed objects within the scene.
- **texture** - the filename of the texture file in case of objects with models with variable texture.
- **VR_model** - the filename of VRML model of the object (string value).

The set of models is indicated between tags <**model_set**> </**model_set**> and has one attribute **name** that corresponds to the name of directory with the models.

5.2 Scene Reconstruction

Predefined object models are inserted into the scene according to the information included in SDF or extracted from ORG log. All the scene reconstruction is based on VRML standard.

The base of this VRML scene is a file that is loaded by the VR module at the start-up. This file contains the script node for connecting animations to the virtual humanoid and the root node of the visual part of the scene. All objects inserted in the scene (including the humanoid) become children of this root node (they are inserted via its 'addChildren' event). Besides that the base file contains six light sources and 'head up display' construction for displaying textual information (typically sections of activity plan).

5.2.1 VRML Objects

Models are stored in separate files (VRML) that include description of object's geometry and, in some cases, even functionality.

The models are implemented as prototypes to allow special VRML events to trigger the functionality. This is the case of objects that are specified as part of another object (see 5.1.3). The model of CD player can be a good example as it implements VRML input event 'ejectButton' to trigger the animation of opening and closing the tray. If the names of these events match the appropriate strings used in activity plan, they can be used to translate the activity plan to VRML description of interaction events.

All the object models implement the fields *translation*, *rotation*, *scale*, *name* and *size*. First three allow the possibility of standard transformations. Name field can identify the object and size is basically the size of its bounding box. The important difference is that the upper face of this "bounding box" is considered the face of the object that can be used to put something on. It is used when determining the height coordinates of undetected objects lying on top or underneath some known objects.

5.2.2 Humanoid

The virtual human that has been used corresponds with the H-Anim 1.1 standard for virtual humanoid [HAn]. It was downloaded from the VRlab website [VRL], where it was available for non-commercial use.

The geometry remained unchanged. From the functional point of view we adapted the model to our needs. We will mention in section 5.3.1 the adapted joint constrains. Besides that we added several Site nodes that mark the end-effectors and attachment points for the manipulated object. The end-effectors are in both hands in the metacarpal region (named *r_hand_tip* and *l_hand_tip*). The attachment points are on the hands too and their location depends on the manipulated object and the type of manipulation (named *r_hand_x* and *l_hand_x*).

5.3 Animation and Interaction

The activity observed by ActIPret system involves human operator performing a manipulation task. Therefore the VR presentation of this activity consists of animation part (motions of the operator) and the interactions (manipulation with the scene). These two parts are closely connected.

The data we can use to show the activity in VR are the trajectory of the hand (from the ORG log) and the activity plan. We use the hand trajectory to animate the virtual humanoid with the inverse kinematics (IK) algorithm. The point that produces the trajectory is basically a centroid of a skin colored blob detected as hand. In our case the IK algorithm cannot animate the joints that are below the wrist in the human joint hierarchy.

We combine the IK animation of kinematic chain of an arm with predefined gestures for finger animation. These gestures are associated with detected actions listed in activity plan. Also the interaction events are associated with these actions and therefore the finger animation is very closely related to resolving the interaction events.

5.3.1 Inverse kinematics

The implemented inverse kinematics (IK) algorithm was an iterative numerical method called Cyclic Coordinate Descent (CCD) [Welman 1993]. The idea of this method is to minimize position and orientation errors by altering one joint variable at a time. Each iteration involves a single traversal of the kinematic chain from the most distal link towards the manipulator base. Each joint variable is modified in order to minimize an objective function.

The CCD is a heuristic iterative algorithm with convergence rate that is a little problematic. On the other hand, a single step of iteration is very simple and the algorithm behaves well in the singular states of kinematic chain. It tends to use joints close to an end-effector rather than spread the motion evenly among all the joints.

For our task of animation operator's arm, this can yield more realistic results even without the use of joint constrains.

The only data for IK that we have is the trajectory - points in the space, no orientation. For this reason, the error function in our implementation of CCD algorithm is simply the distance between current position of end-effector and its destination.

Each joint of the kinematic chain is constrained with the upper and lower rotation limit and another parameter called "stiffness". Both constrains are introduced by H-Anim standard to increase realism of animation generated by IK algorithms. The values of the limits were adopted with the VRML model of human and in some cases adjusted. The "stiffness" parameter should express the willingness of the joint to move and its value was more problematic. The values we have used in 'CD scenario' appeared as a result of "parameter tuning" process. This joint constraint is currently a subject of our research.

We assume that the operator stays at one place when performing the activity. Since the only source of information on operator's position (position or kinematic chain base) is the SDF, this assumption was necessary to implement the IK animation.

5.3.2 Event Model File

Each line of Activity Plan (AP) stands for an action that occurs in the scene. In its 'conceptual language' version, the line is the comma separated list of elements that describe the action. This action is associated with certain hand gesture and interaction between objects in the scene. To convert the AP line to the hand animation and interaction in the scene, we introduced an Event Model File (EMF) the proprietary text file format that describes the conversions. For each action that is expected to appear in the scenario (and in the AP) there must be this Event Model File specified.

The EMF contains three parts that completely describe the event to the VR module. First is the "distance" labelled with <**distance**> tag. It shows how to find event time of occurrence by finding minimum distance between the reference points of two interacting objects over their trajectories. In section 4.2, we mentioned that the time of event known to system is the time of detection and is late behind the trajectories. The objects in question are indicated as indexes of AP elements. Generally it is the <distance> tag followed by the two integer values.

The "hand animation" part in <**animation**> </**animation**> brackets describes the motion of the hand when the event happened. It is the list of relative times and names of files specifying the hang pose (More inc section 5.3.4).

Finally "VRML event description" between <**event**> </**event**> tags shows how to transform the Activity Plan line to the ActIPret VRML description introduced in [Štěpán et al. 2003]. Again the indexes of AP elements are used to indicate the points where that particular element appears in VRML description (More in section 5.3.3).

The 'conceptual language' version of AP line for *pick up* event would be this comma separated list of string values:

```
PICKUP, Hand 1, Object CD, Location undef
```

An example EMF of the *pick up* event:

```
<distance> 1 2

<animation>
-0.3 neutral
-0.1 reachCD
0 holdCD
</animation>

<event>
target_node 2 parent
event_type  MFNode
event_name  removeChildren
event_value 2
next_event
target_node human sites hand_x
event_type  MFNode
event_name  addChildren
event_value 2
</event>
```

The elements number 1 and 2 of this line are *Hand 1* and *CD*, objects with these IDs will be used when processing the interaction event.

5.3.3 Interaction Events

The VR module, as it is based on VRML standard, uses the VRML prototype to describe an interaction event. The Activity Plan line is simply translated to this form.

The lack of the information on event occurrence time was the main problem of the interaction events part of the VR module. Each event involves the hand and another object and it most likely occurs when the hand is close to that object. We find the local minima of distance between hand and object as a likely points of event occurrence. We choose those with the smallest distance that maintain the sequence of events.

We can not choose simply the global minimum for each event because there might be two events that involve the same objects (e.g. the *press button* event).

5.3.4 Hand Animation

With the VRML interpolator-based animation mechanism, we can easily decompose each gesture into sequence of few postures, that are used as key-frames.

To describe these postures we have designed a VRML-based data structure.

```
PROTO Pose [
    exposedField MFString joints []
    exposedField MFRotation rotations []
    exposedField SFVec3f position 0 0 0
    exposedField SFString name ""
]
```

Field *joints* is a list of joints with non zero rotation, *rotations* a list of rotations that belong to these joints. *position* indicates the displacement of whole body.

Although this structure has its origin in the need to compose hand animation as a sequence of postures, it has one more use in the VR module. It is used to describe the starting posture of virtual humanoid's body before the IK algorithm is run to create the animation of body.

The Event Model File specifies how individual postures are queued. Between the <**animation**> </**animation**> tags, there is a sequence of float-string pairs.

The float value indicates the key-frame of the animation as the portion of the interval between two consecutive event occurrences. The string value is the name of the VRML file with the posture description.

As an example we can show the animation part of Event Model File of the pick-up event:

```
<animation>
-0.3 neutral
-0.1 reachCD
0 holdCD
</animation>
```

If previous event happened at the time 0 and current pick-up event is at the time 1, then at the time 0.7 the hand is in neutral position, at the time 0.9 it reaches for CD (hand is open) and at the time of the pick-up event it holds the CD (fingers grasp it).

The names of posture files should be prefixed with *r_* or *l_* for right or left hand performing the action. Thus the EMF entry *neutral* refers to either *r_neutral.wrl* or *l_neutral.wrl* file. The application makes the choice based on the knowledge of end-effector that acts in the particular event.

5.4 Two Handed Scenario

The task observed by the ActIPret system typically involves purposeful motion of both hands thus the interaction events can include both hands. From the functional point of view the side is not important, it is enough to distinguish "one" and "the other" hand. VR module faces a problem of determining the sides.

In case we have trajectories of two hands, we have to follow several conditions. The tracked end-effectors must be properly listed in SDF to define the kinematic chains. The virtual humanoid must contain the end-effectors and possibly the manipulation oriented Site nodes. Most importantly no part of one kinematic chains should be influenced by another. They only can have common base.

All joint, end-effector or any other body part names listed in SDF of EMFs are written in "side-independent" form. That means to exclude the *r_* or *l_* prefix indicating the right of left side. Each tracked end-effector is associated with the side by application. It is done by a simple test which one of the two possibilities is closer to the starting point of the trajectory.

After the tracked end-effectors are associated with end-effectors within humanoid model, the IK solver is called for each kinematic chain to animate the corresponding part of virtual humanoid body. At this point, we assume that the data will not cause conflict. All effectors should be able to reach their destinations as it happened in the observed reality.

6 Conclusions

We were given a task to design a module for virtual reality presentation of results of a cognitive vision project ActIPret. The goal of the project was to create a system that would be able to observe human operator's manipulation activity and decompose it into generalized concepts of actions, thus understand it.

The VR presentation module was far from the main focus, yet it gave us a chance to work with interesting problems of visualizing the human activity using very abstract and incomplete data. We have suggested two possible approaches to the VR presentation of generalized concept of observed activity. The chosen 'instance based' approach uses not only the generalized project output but also some other data available within the project.

Although a functional VR presentation module was created, our work has also another result. We have become familiar with some problems of cooperation in areas that are by their nature quite distant. When seeing the video sequence of human performing an activity, VR and cognitive vision focus on very different things. Due to a low importance of VR presentation within the project framework and the challenging nature of it's main task, we had to do most of the work at the later stages, when the idea of final output had been refined. Thus the ActIPret VR presentation module is an attempt to bridge this wide gap between the areas of research as well as the priorities within the project.

If we should draw some conclusions concerning the success of this attempt, we would have to say that it was not highly successful. The VR presentation did not get over numerous problems and imperfections. It is not comparable to what a state-of-the-art computer animation can be. On the other hand with very limited input it performs well enough to show the activity. The module is quite robust to errors in the activity plan. It can replay the misinterpreted activity and thus it shows the system's errors much better than mere look at the resulting activity plan.

Finally we shall recapitulate the main problems and solutions of this work classified as human animation with insufficient input information. The information we had to use was too abstract often directly in conflict with the clarity of VR visualization. We introduced the interaction event description implemented by EMF file. Human animation part combines IK techniques with key-frames for finger animation taken from EMF. All information related to the particular activity reconstruction gathered from ActIPret system and user input is summarized in SDF file. Finally the reconstructed activity can be stored for easy replay in VRML based format.

Acknowledgements

This work has been supported by the European Union project IST-2001-32184 ActIPret (Interpreting and Understanding Activities of Expert Operators for Teaching and Education). The third author was supported by the project GACR 102/03/0440.

References

ActIPret project home page. `http://actipret.infa.tuwien.ac.at`.

GONCALVES, L. M., KALLMANN, M., AND THALMANN, D. 2001. Programing behaviors with local perception and smart objects: An approach to solve autonomous agents tasks. In *Proceeding of SIGGRAPI 2001*.

H-Anim (Human Animation Working Group). `http://www.hanim.org`.

ŠTĚPÁN, V., ŽÁRA, J., AND HLAVÁČ, V. 2003. Describing human activities for vr presentation. In *VIIP '03: Proceedings of the Visualization, Imaging and Image Processing*, IASTED, Anaheim, USA, 37–42.

Virtual reality lab (former LIG), EPFL Lausanne. `http://vrlab.epfl.ch`.

WELMAN, C. 1993. *Kinematics and Geometric Constraints for Articulated Figure Manipulation*. Master's thesis, Simon Fraser University.

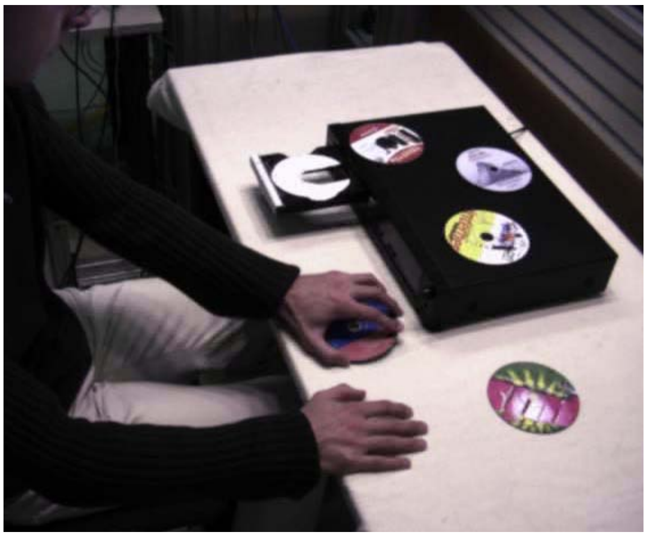

Figure 3: A frame from a real footage taken by one of ActIpret system cameras (the two handed ‘CD scenario’ activity).

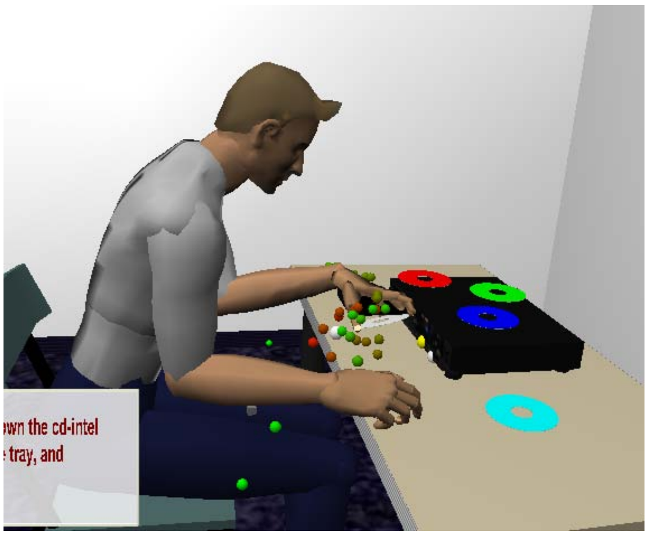

Figure 5: A screenshot of the activity reconstructed in VR. Small green and red spheres indicate the hand trajectory.

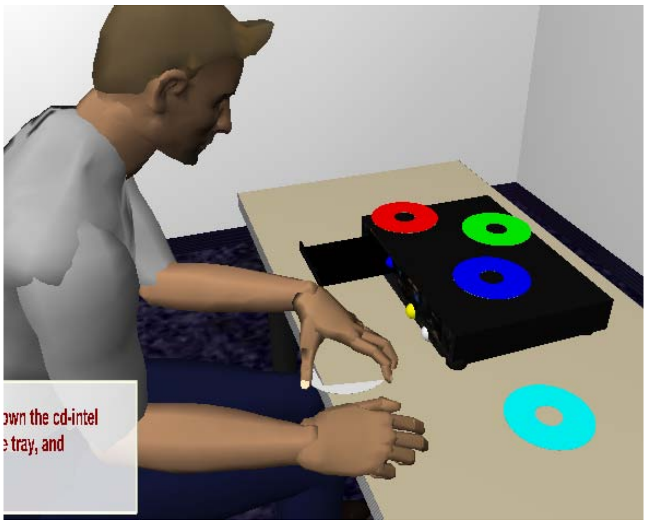

Figure 4: This figure shows a collision error.The hand with CD collides with the desk. The generic gestures combined with specific trajectory often result in such collisions.

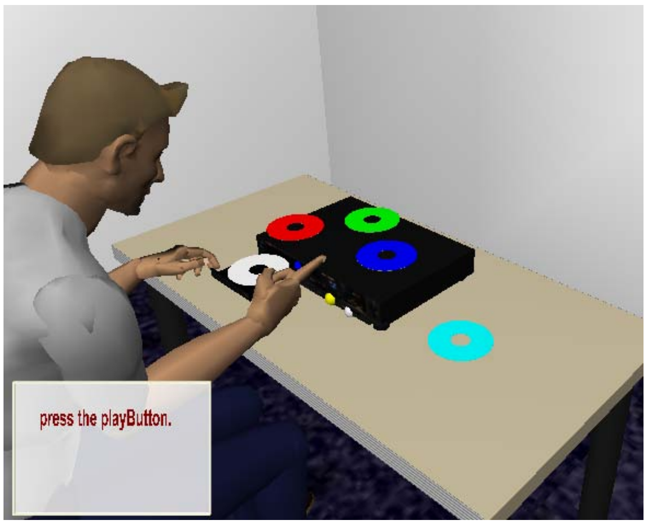

Figure 6: Another screenshot of reconstructed activity. Note the textual “hints” displayed at the left lower corner.