Autonomous Agents Interacting With Their Virtual Environment Through Synoptic Objects

Abstract

We describe in this paper the STARFISH (Synoptic-objects for Tracking Actions Received From Interactive Surfaces and Humanoids) architecture that uses Synoptic Objects to allow real-time object manipulation by autonomous agents in an informed environment. We define a minimal set of primitive Basic Actions which are used to build Complex Actions. We then assign these actions to Interactive Surfaces which are the parts of an object's geometry that are concerned by the action. The agent then uses these Interactive Surfaces to get the data specific to the object when it wants to manipulate it and to adapt its behavior accordingly. We will show how to model a Synoptic Object and assign an internal state machine to it that allows it to track the progress of the actions. We finally illustrate our approach with a detailed example of an agent opening a door.

Keywords: Virtual Humans, Autonomous Agents, Informed Environment, Object Modeling, Object Interaction

1   Introduction

When populating a virtual world with synthetic actors, one of the problems that arise is how to make these actors manipulate the objects which surround them. Two approaches are usually taken to solve this problem: the motion capture approach where all actions on all objects are pre-recorded and replayed at will, and a more flexible approach where information is stored into the virtual world, which is then referred to as an informed environment [1], to give the actors information on how to proceed.

Our goal, is to create an informed environment which contains enough information to allow autonomous agents to interact with it but with enough flexibility as to allow a certain freedom and laxness when the interaction occurs. In this paper, we first give an overview of the different approaches to this problem and then briefly present the Synoptic-objects for Tracking Actions Received From Interactive Surfaces and Humanoids (STARFISH) system. We will first present the STARFISH Actions and Interactive Surfaces which are the basic components of every Synoptic Object and then focus on the Synoptic Objects, which are at the heart of STARFISH.

We will finally demonstrate how these objects allow us to create believable interactions between autonomous agents and their surroundings.

2   Related work

Examples of motion capture interaction can be seen in most video games nowadays. A good showcase of such an approach is the game *The Sims* [2]. All possible interactions with all possible objects are recorded using motion capture and then replayed as is, completely unaltered, whenever needed. This gives extremely realistic animations since it replays motions performed by a live human being. But this realism comes at the price of having the same animation and the same behavior repeated whenever an action is performed; and this, in the long run, tends to be unrealistic since real humans never perform the same action twice in exactly the same way.

A hybrid approach is used by Badler et al. in the Parameterized Action Representation (PAR) system [3], where objects are integral parts in defining the action to be taken. An object specific reasoning (OSR) module [4] addresses the agent-object interaction issue. The OSR keeps some interaction information for each object, such as graspable site, hand shape and approach direction that are sufficient for deciding and performing grasping tasks, but not for interacting with more complex objects. The gestures accomplished by an agent during interaction are completely synthetic, and created by using the EMOTE model [5] which is an interpretation of the Laban Movement Analysis. Thanks to that it is possible to parameterize the gestures in a way that affects their outcome and thus create different behaviors adapted to different situations. But the PAR itself is entirely pre-specified and thus needs to be re-adapted offline if the targeted object is to be changed. We qualify this approach as hybrid because, even though there is no information in the environment itself, the actions to be taken depend directly on the objects they are applied to.

As for informed environments, one such environment is STEVE [6]. In STEVE, some basic information is stored inside the objects, to allow some manipulations such as pushing a button or pulling a lever. But the tasks STEVE is able to perform are limited and very domain specific. It is used for procedural training in virtual reality for maintenance tasks

aboard a ship and is not adapted to a more general everyday environment. A more generic approach is taken by Kallmann [7, 8] with the Smart Object architecture. All the information necessary to interact with the object is contained in the object itself. A Smart Object which can hold a relatively complex action, will instruct the synthetic actor on the actions to do step by step. The architecture uses a basic set of primitive gestures which are modified in real time using inverse kinematics to adapt the agent's motion to the position of the object it is interacting with. All the information relative to the interaction is stored in the objects themselves and the agent becomes nothing more than an empty shell animated by the object. This is, in our opinion, contrary to the notion of *autonomous* agents.

There is also a top-down approach which consists in using image recognition, planning and learning in order to dynamically build and create manipulation data for agent-object interaction. To our knowledge, such an approach has not yet been implemented for real-time virtual human simulations and has only been used in robotics. And it is our opinion that, if such an approach were to be used, it would still not be able to solve the problem of interacting with complex machines. This kind of interaction would necessitate knowledge not readily available by this technique, such as functionality and inner workings.

3 STARFISH

STARFISH, is a new system which allows the easy definition of interactions between autonomous agents and the objects in the environment.

At the heart of STARFISH are Synoptic Objects which are objects containing information describing the way they can be interacted with. This information is made available through two main components:

STARFISH Actions which can be broken down into two subparts:

- Basic Actions which are a minimal set of primitive atomic actions.
- Complex Actions which are built by composing Basic Actions.

Interactive Surfaces which are usually parts of an object's surface that act like hotspots during interaction.

We will do a brief introduction of these components as they are essential in understanding the workings of the Synoptic Objects. We will also present the whiteboard, which allows the communication between the agents and the objects. Then, in Section 4 we will detail the Synoptic Object concept which is the main topic of this paper.

3.1 STARFISH Actions

The first category of STARFISH Actions consists of a group of simple atomic actions, the Basic Actions which are the building blocks used to create the Complex Actions. A similar approach is used in Brahms [9, 10], a domain general, agent oriented language used to simulate everyday activities. It uses *Primitive Activities* to create behaviors that are used to model complex work practice scenarios. The set of primitive

activites used in Brahms is not a closed one, and can be added to and enriched constantly. This does create a variety for the definition of complex behaviors but it requires to completely define a primitive activity whenever one is introduced in the system, and readapt the system to work with it.

Inspired by Schank's *Theory of Conceptual Dependency* [11], Basic Actions are an extension of the theory's primitive ACTs. Schank says that *in order to create conceptual structures that will uniquely and unambiguously represent the meaning of an utterance, it is necessary to establish 'primitive' underlying actions and states into which verbs can be mapped* [12]. These *primitive underlying actions* is what he called the primitive ACTs. Since Schank was able to deconstruct all sentences in common English into these ACTs, why not take these ACTs and use them to generate more complex actions? We use a subset of Schank's ACTs that allow us to define any action that has an impact on the physical world. Through the Basic Actions described in Table 1, we can build more complex and varied actions. We have omitted the ACTs relating to mental processes such as thinking and decision making since STARFISH is not yet capable of managing such processes. Note that for the remainder of this paper, whenever we refer to a Basic Action, its name will be typeset in **bold face** to differentiate it from the corresponding verb.

Give	Transfer a relationship (give, take)
Transfer	Transfer location of an object (go, carry)
Displace	Apply force to an object (throw)
Move	Move own body part (kick, reach)
Grasp	Grab an object (grasp)
Ingest	Take an object into own body (eat)
Speak	Produce sound (say, sing)
Attend	Focus sense organ (listen, look at)

Table 1: The complete set of Basic Actions.

For example, the *Open Door* action can be easily decomposed into its Basic Actions:

- **Transfer** self to the door.
- **Move** arm towards knob.
- **Grasp** knob.
- **Move** hand to turn knob.
- **Move** arm to open door.
- un-**Grasp** to let go of the knob.

It is possible to specify actions which have to be executed in parallel such as **Attend**-ing the agents eyes to its arm (and eventually **Move**-ing its head) as it is **Move**-ing its arm towards the knob. It is also important to note that a Basic Action can, by itself, be used as a full fledged STARFISH Action: the action of kicking a can for example only consists in **Move**-ing the agent's leg until it comes into contact with the can.

The action is thus completely defined. The agent then uses the Interactive Surfaces of the door to take the necessary parameters it needs to actually compute the path of its arm and hand.

3.2 Interactive Surfaces

Interactive Surfaces are used to model and describe the affordances of an object, a term coined by Gibson in his *Theory of Affordances* [13]. Quoting Gibson's own definition, *the affordances of the environment are what it offers the animal, what it provides or furnishes.* The *animal* is, in our case, the autonomous agent and the *environment* an informed virtual one.

Interactive Surfaces are generally parts of the object's geometry that act as hotspots when a certain action is to be accomplished, such as where to place the agent's hand when **Grasp**ing the doorknob. Each Synoptic Object can have as many Interactive Surfaces as needed, depending on how much interaction it offers.

Interactive Surfaces can also be used to define the positioning of the agent relatively to the Synoptic Object it wants to interact with. For example, an Interactive Surface might be used to indicate a relative position where the agent should be when it **Transfer**s itself toward a Synoptic Object. In the previous example, when the agent decides to open the door, it uses the Interactive Surface corresponding to the **Transfer** action, to place itself in a correct position to be able to continue to the next step in the *Open Door* STARFISH Action. Since it is possible to execute Basic Actions in parallel, the **Transfer** Interactive Surface can also be used as a constraint for the agent: when the door is being opened, all its Interactive Surfaces are moving with it. By constraining the agent to stay in the appropriate area by **Transfer**-ing into the Interactive Surface, it will be able to stay out of the path of the door.

In addition to defining a spatial area concerned by a corresponding action, Interactive Surfaces also contain information to quantify the path and trajectory of that action. The actual animation of the agent is done by using [left blank] [14], a motion adaptation system. Going back to the Basic Action consisting of turning the knob to open the door, the Interactive Surface used to perform that action tells the agent which way the knob should be turned. Since we use a motion adaptation system, the actual data contained in the Interactive Surface is just an indication of what the action should consist of and does not, in any way, define a complete animation. This goes a long way to simplify the definition of an Interactive Surface and also gives the possibility to the autonomous agent to adapt its movements depending on body and limb position. In other words, the way the agent moves to perform a specific action always depends on where that agent is and where its arm is positioned when it is asked to interact with an object. This helps in reducing the effect of robot like behaviors.

The use of an Interactive Surface to indicate a general area for interaction between a Synoptic Object and a virtual humanoid combined with the motion adaptation system, adds a relatively nondeterministic element to the simulation which

enhances its realism. Another advantage in using the motion adaptation system, is that it adapts the animation to the agent's morphology. The data that complements the Interactive Surface is thus valid for whatever agent wants to use the corresponding Synoptic Object and completely independent of any physical constraints the agent might have. Through STARFISH the agent can get the information needed to interact with a Synoptic Object and then decides for itself whether it can actually perform that action.

3.3 The Whiteboard

The purpose of the STARFISH whiteboard is to simply bring everything together. It keeps track of the physical location in 3D space of all Synoptic Objects present in the simulation. It acts as a database and is queried whenever interaction information is needed. All communications between agents and Synoptic Objects are initiated through the whiteboard. The agent first asks the whiteboard whether it can start to interact with a certain object. If the interaction is possible, the whiteboard informs the agent about the object in question and all further communication is done between the agent and the object directly. Message passing is done through the [left blank] [15], our virtual reality simulation platform.
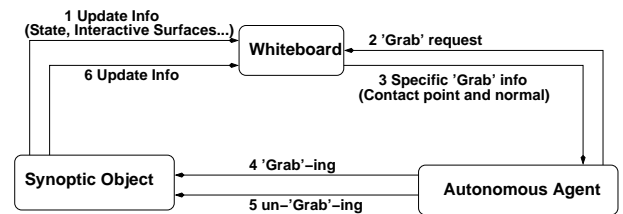

Figure 1: Typical process for a **Grab** action.

Since the whiteboard is only solicited whenever an interaction is initiated, it is not necessary to update it in real time. It is initialized at the start of the simulation and then explicitly refreshed whenever an interaction modifies the state of a Synoptic Object.

The arrows in Figure 1 represent messages sent between the different modules, and the numbers show the order in which these events are sent. As can be seen, the whiteboard is only updated by the Synoptic Object at load time and just after the STARFISH Action is done.

4 Synoptic Objects

A Synoptic Object is an object designed to offer to the autonomous agent a summary, or synopsis, of what interactions it affords. When an agent queries such an object, it knows what actions it can perform, where it should position itself, where it should place its hands, what state the object is in, whether it is allowed to perform the action, etc. . . All these indications, are given through the use of Interactive Surfaces and STARFISH Actions.

4.1 Modeling a Synoptic Object

When a Synoptic Object is being modeled, the first step is to specify the areas of the object's surface we want to make interactive and assign them some information in order to create

an Interactive Surface. A single Interactive Surface can contain multiple independent surfaces which are not necessarily contiguous. The definition of an Interactive Surface is done through the STARFISH editor: the user basically selects a 2D area in the editor's window which is then projected into 3D space onto the object itself. It is then possible to refine and modify the projected surface. Once the geometric shape of the Interactive Surface is satisfactory, the user can add some properties such as the Basic Action it is associated to and the parameters necessary to allow the correct execution of that action. Figure 2 shows two different Interactive Surfaces associated to a door. The first one is associated to the **Grab** action and is placed on the knob. The second one is adjacent to the door and is used by the **Transfer** action to correctly position the agent during the interaction.
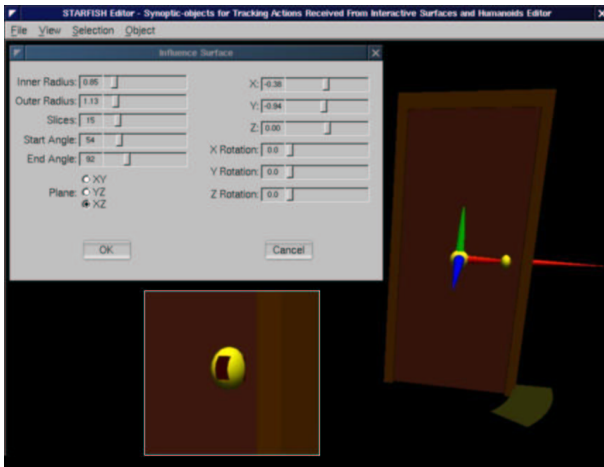

Figure 2: Modeling a door in the STARFISH Editor.

An Interactive Surface of a Synoptic Object can be used by more than one Basic Action. If we turn again to the action of opening a door, the information on where to **Grasp** the knob, which way to **Move** the hand to open it, and also in which direction to **Move** the arm to open the door (whether to push, pull or slide) is contained in a single Interactive Surface which is the surface of the doorknob itself.

On the other hand, a same Basic Action can be attached to more than one Interactive Surface depending on what STARFISH Action it is part of. If we take the example of a suitcase, the agent will **Grasp** the case by the handle if it wants to carry the case, by the lock if it wants to unlock it, and by somewhere along the outer surface if it wants to open it. The selection of the appropriate Interactive Surface is done through the state machine assigned to the object in the next step.

The second stage consists in defining the behavior a Synoptic Object by assigning a state machine to it. And this is where the term *synoptic* comes from. Through the state machine, the Synoptic Object *knows* what state it is in, what actions are allowed when it is in this state, and what states are attainable from the current state. Transitions between states

are defined through Basic Actions and the Interactive Surfaces associated with them. There are also groups of states we call *paths*, which correspond to the execution of the multiple Basic Actions compounding a STARFISH Action. The transition to the first state of such a path has one more parameter which is the STARFISH Action it corresponds to. Once the state machine is engaged in a path, the transitions from state to state inside this path must be followed through to the end to reach a stable state from which the Synoptic Object can safely wait for other requests. When an agent decides to interact with a Synoptic Object, it first queries the object to know whether it is possible for it to perform the required action. In other words, if the agent wants to open a door, it *tells* the door that it wants to open it and the door either replies that it is already open or tells the agent which STARFISH Action to perform in order for the door to be *open*.
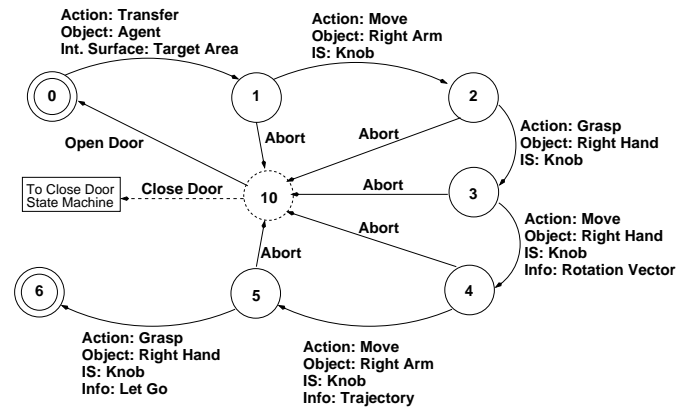

Figure 3: State machine of the *open door* action.

Even though it is necessary to follow a path in order to successfully accomplish a STARFISH Action, this does not imply that a path must be followed through to the end. Figure 3 shows the state machin describing the path of the *open door* action from the stable state *closed* (state 0), to the stable state *open* (state 6). As can be seen in in teh figure, it is possible to abort the *open door* action at any time. This will result in putting the door into a state (marked 10) which is neither *open* nor *closed*. The next agent to interact with the door will then know that both the *open* and *close* actions are valid and that they will both eventually result in putting the Synoptic Object into a stable state.

The state machines are controlled by [left blank] [16] which is an extension of the [left blank] System [17]. The latter is a system that allows the running of hierarchical and parallel state machines. It has been extended into [left blank] which introduces the notions of priority and resources. With the extension, it becomes possible to choose which state to transit to, depending on the priority assigned to it and the resources that state uses. For example, if a door is locked and the *key* resource is not available, it will never be able to transit into the *open* state and thus the *open door* action becomes impossible to achieve without the proper resource.

4

Figure 3 shows an example of a very basic state machine. The transition from one state to another is only done when the Basic Action describing that transition is successfully completed. This means that the Synoptic Object will be in state 2 *after* the agent's hand is successfully on the doorknob. The completion of the action is detected by the agent which has enough information to know that it has successfully accomplished the required task (e.g. the hand is touching the knob). It then informs the Synoptic Object which transits into the next state and gives the agent the data it requires to complete the next step.

Seeing how both these stages are completely independent, we don't have to go through with them both when creating a new Synoptic Object. When an object with a new shape but with the same functionality has to be introduced into the simulation, it is only necessary to model its appropriate Interactive Surfaces without having to modify their associated behaviors. Suppose we already have a door with its Interactive Surfaces and state machines already defined. If we need to use a new door which opens in the same way, we only need to modify the geometry of the Interactive Surfaces associated with it so they are adapted to the geometry of the new door. No further processing for the agent or the Synoptic Object is necessary. In short, when we modify the geometry of an object, we do not need to modify or readapt any of its behaviors. If the geometry of the Interactive Surfaces reflects the geometry of the Synoptic Object they belong to, the change will be transparent to the agent which will still behave accordingly.

4.2 Interacting with a Synoptic Object, an example

We will now describe how interaction occurs between an autonomous agent and a Synoptic Object during a real-time simulation. For this, we will go into the details of the *Open Door* STARFISH Action we mentioned in section 3.1.

We will not go into the description of the decision cycle of the agent since it is out of the scope of this paper. Instead we will directly start the description after the decision is made, which means that the agent now wants to open the closed door that is in front of it. The agent knows the Basic Actions that constitute the STARFISH Action that will eventually lead to the door being in the *open* state. Also note that the communication between the agent and the Synoptic Object is done through the whiteboard. This example is illustrated in Figure 4.

The first thing the agent does, is ask the door whether it can be opened by the agent. The synoptic door object first checks to see if it is in a state where the *open door* action can be performed. It then checks to see if no other agent has already requested that action and is currently performing it. If that is the case, it is possible to ask the other agent to abort the action. After that, the agent receives the permission to open the door and is given the Interactive Surface corresponding to the **Transfer** Basic Action. Having its destination, the agent then **Transfer**s itself towards the Interactive Surface it just received by giving the Interactive Surface's coordinates to a navigation algorithm [18].

Once the destination is reached, the agent informs the synoptic door object that it wants to proceed to the next step.

Figure 4: An agent pulling open a door.

The door then gives the agent the geometry of the Interactive Surface of the handle and its coordinates in space. The agent then proceeds by estimating a path for its arm to **Move** from its current position towards the knob. It does so by using the SWIFT++ [19] collision detection engine which is, is capable of detecting collisions between non convex objects. This important SWIFT++ property allows us to remove any restrictions on the shapes of the Interactive Surfaces that belong to an object. Having performed the estimation, the agent is able to calculate a contact point and contact normal between its hand and the knob. It then uses this information and passes it to the motion adaptation engine which generates the movement of the arm towards the knob.

Then the agent **Grasp**s the knob. The automatic generation of a grasping motion [20] being a very complex field and out of the scope of our current work, we use an estimation of the way the agent's fingers should be placed.

The agent then receives information from the synoptic door object on the direction in which it should **Move** its hand in order to turn the knob and unlatch it. This information, in the form of a rotation vector, is transmitted to the motion adaptation engine which twists the hand accordingly.

Depending on the way the door opens, different kinds of information can be contained in this step. If the door opens by sliding, a simple direction vector is enough to calculate the arm's trajectory. If it opens by pulling or pushing it is better, for the time being, to give the complete arm trajectory to the agent which will proceed to **Move** its arm accordingly. In the future, a simple direction vector, outward for pulling and inward for pushing, should be enough to be able to determine the trajectory of the arm. But this is a work in progress.

The final step is letting go of the doorknob once the door is completely open. The agent simply unflexes its fingers to un-**Grasp** the handle, and brings its arm back to an idle position.

5

The door is now open.

5 Conclusion

We have presented an architecture capable of informing an autonomous agent in a virtual world about its surroundings. The resulting informed environment is rich enough to offer many possibilities for the agent even though it is only based on eight Basic Actions. The versatility of these Basic Actions allows the building of almost any complex action and thus enables the agent to perform virtually any interaction task. Furthermore, the definition of these complex STARFISH Actions is easy since it consists of simple geometric Interactive Surfaces completed by little more movement specific information. This allows the agent to adapt the interaction behavior to itself instead of adapting itself to the behavior.

We have shown in our paper how STARFISH is used in a virtual reality simulation environment to create behaviors and animations. Future work will focus on the extension of STARFISH in order to provide the necessary information needed to allow reasoning about object manipulation ans interaction.

References

[1] N. Farenc, R. Boulic, and D. Thalman. An informed environment dedicated to the simulation of virtual humans in urban context. In *Eurographics 1999*, volume 18, 1999.

[2] Electronic Arts and Maxis Entertainment. The sims. http://thesims.ea.com/, 1999.

[3] N.I. Badler, R. Bindiganavale, J. Allbeck, W. Schuler, L. Zhao, S.J. Lee, H. Shin, and M. Palmer. Parameterized action representation and natural language instructions for dynamic behavior modification of embodied agents. In *AAAI Spring Symposium 2000*, 2000.

[4] L. Levison and N. Badler. How animated agents perform tasks: Connecting planning and manipulation through object-specific reasoning. In *AAAI'94 Spring Symposium: Toward Physical Interaction and Manipulation*, 1994.

[5] M. Costa, L. Zhao, D. Chi, and N.I. Badler. The emote model for effort and shape. In *SIGGRAPH 2000*, 2000.

[6] J. Rickel and W.L. Johnson. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13:343–382, 1999.

[7] M. Kallmann. *Object Interaction in Real-Time Virtual Environments*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2001.

[8] M. Kallmann and D. Thalmann. Modeling objects for interaction tasks. In *EGCAS'98 - 9th Eurographics Workshop on Animation and Simulation*, 1998.

[9] M. Sierhuis. *Modeling and Simulating Work Practice*. PhD thesis, University of Amsterdam, 2001.

[10] M. Sierhuis, W.J. Clancey, C. Seah, J.P. Trimble, and M.H. Sims. Modeling and simulation for mission operations work system design. *Journal of Management Information Systems*, 19(4):85–128, 2003.

[11] R.C. Schank. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3:532–631, 1972.

[12] R.C. Schank, N. Goldman, C.J. Rieger, and C.K. Riesbeck. Primitive concepts underlying verbs of thought. Internal Memo of the Stanford Artificial Intelligence Project, February 1972.

[13] J.J. Gibson. *The Ecological Approach to Visual Perception*, chapter 8: The Theory of Affordances, pages 127–143. Lawrence Erlbaum Associates, 1986.

[14] To be given in the final paper, 2004.

[15] To be given in the final paper, 2002.

[16] To be given in the final paper, 2002.

[17] To be given in the final paper, 2001.

[18] To be given in the final paper, 2004.

[19] S. Ehmann. and M. Lin. Accurate and fast proximity queries between polyhedra using surface decomposition. In *Eurographics 2001*, 2001.

[20] R. Mas and D. Thalmann. A hand control and automatic grasping system for synthetic actors. In *Eurographics '94*, 1994.