A Virtual Reality Platform for Modeling Cognitive Development

Hector Jasso
Department of Computer Science and Engineering
UC San Diego, USA
hjasso@cs.ucsd.edu

Jochen Triesch
Cognitive Science Department
UC San Diego, USA
triesch@cogsci.ucsd.edu

Abstract

We present a virtual reality platform for developing and evaluating embodied models of cognitive development. The platform facilitates structuring of the learning agent, of its visual environment, and of other virtual characters that interact with the learning agent. It allows to systematically study the role of the visual and social environment for the development of particular cognitive skills in a controlled fashion. We describe how it is currently being used for constructing an embodied model of the emergence of gaze following in infant-caregiver interactions and discuss the relative benefits of virtual vs. robotic modeling approaches.

1. Introduction

Recently, the field of cognitive science has been paying close attention to the fact that cognitive skills are unlikely to be fully specified genetically, but develop through interactions with the environment and caregivers. The importance of interactions with the physical and social environment for cognitive development has been stressed by connectionist [7] and dynamical systems [17] approaches.

Developmental schemes are also being proposed in the field of intelligent robotics [1, 3, 18]. Instead of building a fully working robot, a body capable of interacting with the environment is given general learning mechanisms that allow it to evaluate the results of its actions. It is then "set free" in the world to learn a task through repeated interactions with both the environment and a human supervisor.

Our motivation is to develop embodied models of cognitive development, that allow to systematically study the emergence of cognitive skills in naturalistic settings. We focus on visually mediated skills since vision is the dominant modality for humans. The kinds of cognitive skills whose development we would ultimately like to model range from gaze and point following and other shared attention skills over imitation of complex behaviors to language acquisition. Our hope is that embodied computational models will help to clarify the mechanisms underlying the emergence of cognitive skills and elucidate the role of intrinsic and environmental factors in this development.

In this paper, we present a platform for creating embodied computational models of the emergence of cognitive skills using computer-generated *virtual environments*. These virtual environments allow the semi-realistic rendering of arbitrary visual surroundings that make it easy to relate model simulations to experimental data gathered in various settings. Our platform facilitates structuring of the graphical environment and of any social agents in the model. Typically, a single developing infant and a single caregiver are modeled, but arbitrary physical and social settings are easily accommodated. To illustrate the features of our our platform, we show how it can be used to build an embodied model of the emergence of gaze following in infant-caregiver interactions. This effort is a component of a larger research project studying the emergence of shared attention skills within the MESA (Modeling the Emergence of Shared Attention) project at the University of California San Diego[1].

The remainder of the paper is organized as follows. Section 2 describes our modeling platform and the underlying software infrastructure. Section 3 shows how it is currently being used to build an embodied model of the emergence of gaze following in mother infant interactions. Finally, we discuss our work and the relative benefits of virtual vs.

[1] `http://mesa.ucsd.edu`

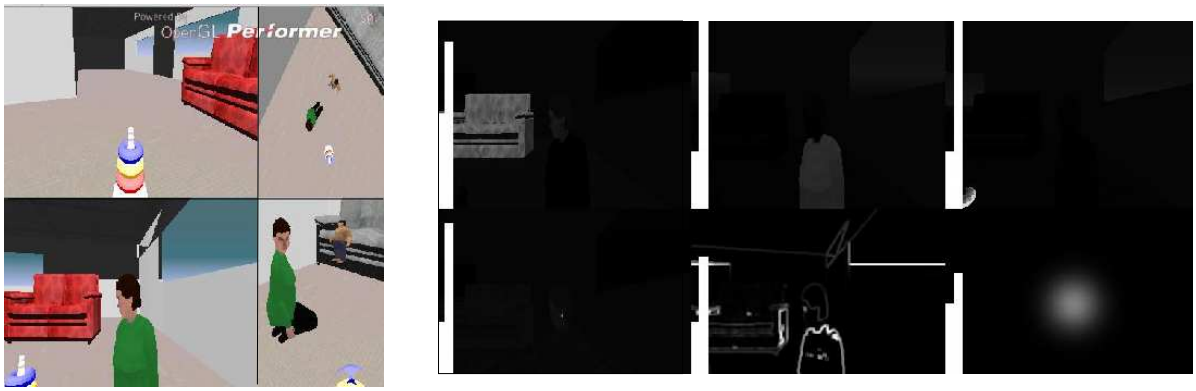

Figure 1. Left: various views of a virtual living room used to model the emergence of gaze following. From top left, clockwise: caregiver's view, birds eye view, lateral view, and infant's view. Right: Saliency maps generated by analyzing the infant's visual input (lower left image in left half of figure). Top row, left to right: red, green, blue. Bottom row, left to right: yellow, contrast, face position. Bars on left of each saliency map indicate the intrinsic reward of this feature and the current habituation level.

robotic modeling approaches in Section 4.

2. The Platform

2.1. Platform Overview

The platform allows the construction of semi-realistic models of arbitrary visual environments. A virtual room with furniture and objects can be set up easily to model, say, a testing room used in a controlled developmental psychology experiment, or a typical living room. These visual environments are populated with virtual characters. The behavior and learning mechanisms of all characters can be specified. Typically, a virtual character will have a vision system that receives images from a virtual camera placed inside the character's head. The simulated vision system will process these images and the resulting representation will drive the character's behavior [15]. Figure 1 shows an example setting.

An overview of the software structure is given in Figure 2. The central core of software, the "Simulation Environment," is responsible for simulating the learning agent (infant model) and its social and physical environment (caregiver model, objects, ...). The Simulation Environment was programmed in C++ and will be described in more detail below. It interfaces with a number of 3rd party libraries for animating human characters (BDI DI-Guy), managing and rendering of the graphics (SGI OpenGL Performer), and visual processing of rendered images to simulate the agents' vision systems (OpenCV).

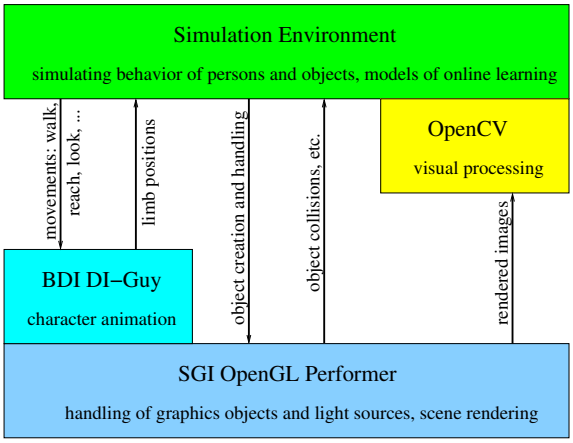

Figure 2. Overview of software structure.

The platform currently runs on a Dell Dimension 4600 desktop computer with a Pentium 4 processor running at 2.8GHz. The operating system is Linux. An NVidia GeForce video graphics accelerator speeds up the graphical simulations.

2.2. Third Party Software Libraries

OpenGL Performer. The Silicon Graphics *OpenGL Performer*[2] toolkit is used to create the graphical environment for running the experiments. OpenGL Performer is

[2] `http://www.sgi.com/products/software/performer/`

a programming interface built atop the industry standard *OpenGL* graphics library . It can import textured 3D objects in many formats, including OpenFlight (.flt extension) and 3D Studio Max (.3ds extension). OpenGL is a software interface for graphics hardware that allows the production of high-quality color images of 3D objects. It can be used to build geometric models, view them interactively in 3D, and perform operations like texture mapping and depth cueing. It can be used to manipulate lighting conditions, introduce fog, do motion blur, perform specular lighting, and other visual manipulations. It also provides virtual cameras that can be positioned at any location to view the simulated world.

DI-Guy. On top of OpenGL Performer, Boston Dynamics's *DI-Guy* libraries[3] provide lifelike human characters that can be created and readily inserted into the virtual world. They can be controlled using simple high-level commands such as "look at position (X, Y, Z)," or "reach for position (X, Y, Z) using the left arm," resulting in smooth and lifelike movements being generated automatically. The facial expression of characters can be queried and modified. DI-Guy provides access to the character's coordinates and link positions such as arm and leg segments, shoulders, hips, head, etc. More than 800 different functions for manipulating and querying the characters are available in all. Male and female characters of different ages are available, configurable with different appearances such as clothing style.

OpenCV. Querying the position of a character's head allows us to dynamically position a virtual camera at the same location, thus accessing the character's point of view. The images coming from the camera can be processed using Intel's *OpenCV* library[4] of optimized visual processing routines. OpenCV is an open-source, extendable software intended for real-time computer vision, and is useful for object tracking, segmentation, and recognition, face and gesture recognition, motion understanding, and mobile robotics. It provides routines for image processing such as contour processing, line and ellipse fitting, convex hull calculation, and calculation of various image statistics.

2.3. The Simulation Environment

The Simulation Environment comprises a number of classes to facilitate the creation and running of simulations. Following is a description of the most important ones.

The Object Class. The OBJECT class is used to create all inanimate objects (walls, furniture, toys, etc.) in the simulation. Instances of the OBJECT class are created by giving the name of the file containing the description of a 3D geometrically modeled object, a name to be used as a handle, a boolean variable stating whether the object should be allowed to move, and its initial scale. The file must be of a format readable by OpenGL Performer, such as 3D Studio Max (.3ds files) or OpenFlight (.flt files). When an OBJECT is created, it is attached to the Performer environment. There are methods for changing the position of the OBJECT, for rotating it, and changing its scale. Thus, it can easily be modeled that characters in the simulation can grasp and manipulate objects, if this is desired.

The Object Manager Class. The OBJECT MANAGER class holds an array of instances of the OBJECT class. The OBJECT MANAGER has methods for adding objects (which must be previously created) to the scene, removing them, and querying their visibility from a specific location. The latter function allows to assess if, e.g., an object is within the field of view of a character, or if the character is looking directly at an object.

The Person Class. The PERSON class is used to add any characters to the simulation. These may be rather complicated models of, say, a developing infant simulating its visual perception and learning processes, or they may be rather simplistic agents that behave according to simple scripts. To create an instance of the PERSON class, a DI-Guy character type must be specified, which determines the visual appearance of the person, along with a handle to the OpenGL Performer camera assigned to the character. The BRAIN type and VISION SYSTEM type (see below) must be specified. If the character's actions will result from a script, then a filename with the script must be given. For example, such a script may specify what the character is looking at at any given time. One BRAIN object and one VISION SYSTEM object are created, according to the parameters passed when creating the PERSON object. The PERSON object must be called periodically using the "update" method. This causes the link corresponding to the head of the character to be queried, and its coordinates to be passed to the virtual camera associated with the character. The image from the virtual camera in turn is passed to the character's VISION SYSTEM, if the character has any. The output of the VISION SYSTEM along with a handle to the DI-Guy character is passed to the BRAIN object, which will decide the next action to take and execute it in the DI-Guy character.

The Brain class. The BRAIN class specifies the actions to be taken by an instance of the PERSON class. The space of allowable actions is determined by the DI-Guy character type associated with the person. The simplest way of how a BRAIN object can control the actions of a PERSON is by following a script. In this case the PERSON will "play back" a pre-specified sequence of actions like a tape recorder. More interestingly, a BRAIN object can contain a simulation of the person's nervous system (at various levels of abstraction). The only constraint is that this simulation has to run in discrete time steps. For example, the BRAIN object may

[3] `http://www.bdi.com`

[4] `http://www.intel.com/research/mrl/research/opencv/`

instantiate a reinforcement learning agent [14] whose state information is derived from a perceptual process (see below) and whose action space is the space of allowable actions for this character. An "update" method is called every time step to do any perceptual processing, generate new actions, and possibly simulate experience dependent learning.

The actions used to control a character are fairly high-level commands such as "look to location (X,Y,Z)," "walk in direction Θ with speed v," or "reach for location (X,Y,Z) with the left arm," compared to direct specification of joint angles or torques. Thus, this simulation platform is not well suited for studying the development of such motor behaviors. Our focus is on the development of higher-level skills that use gaze shifts, reaches, etc. as building blocks. Thus, it is assumed that elementary behaviors such as looking and reaching have already developed and can be executed reliably in the age group of infants being modeled — an assumption that of course needs to be verified for the particular skills and ages under consideration. The positive aspect of this is that it allows to focus efforts on modeling the development of higher level cognitive processes without having to worry about such lower-level skills. This is in sharp contrast to robotic models of infant development, where invariably a significant portion of time is spent on implementing such lower level skills. In fact, skills like two-legged walking and running, or reaching and grasping are still full-blown research topics in their own right in the area of humanoid robotics.

The Vision System class. The VISION SYSTEM class specifies the processing to be done on the raw image corresponding to the person's point of view (as extracted from a virtual camera dynamically positioned inside the person's head). It is used to construct a representation of the visual scene that a BRAIN object can use to generate behavior. Thus, it will typically contain various computer vision algorithms and/or some more specific models of visual processing in human infants, depending on the primary goal of the model.

If desirable, the VISION SYSTEM class may also use so-called "oracle vision" to speed up the simulation. Since the simulation environment provides perfect knowledge about the state of all objects and characters in the simulation, it is sometimes neither necessary nor desirable to infer such knowledge from the rendered images through computer vision techniques, which can be difficult and time consuming. Instead, some property, say the identity of an object in the field of view, can simply be looked up in the internal representations maintained by the simulation environment — it functions as an oracle. This simplification is desirable if the visual processing (in this case object recognition) is not central to the developmental process under consideration, and if it can be assumed that it is sufficiently well developed prior to the developmental process being studied primarily. In contrast, in a robotic model of infant development, there is no "oracle" available, which means that all perceptual processes required for the cognitive skill under consideration have to be modeled explicitly. This is time-consuming and difficult.

Main Program and Control Flow. The main program is written in C++ using object-oriented programming. OpenGL Performer is first initialized, and a scene with a light source is created and positioned. A window to display the 3D world is initialized, and positioned on the screen. Virtual cameras are created and positioned in the world, for example as a birds eye view or a lateral view. Cameras corresponding to the characters are created but positioned dynamically as the characters move their heads. Each camera's field of view can be set (characters would usually have around a 90^o field of view), and can be configured to eliminate objects that are too close or too far. All cameras created are linked to the window that displays the 3D world. Environment settings such as fog, clouds, etc. can be specified. The DI-Guy platform is then initialized, and a scenario is created. The scenario holds information about all the characters, and must be used to create new characters. New instances of the PERSON class are created, and their activities are specified by periodically giving them new actions to perform. The level of graphical detail of the characters can be specified to either get fairly realistically looking characters or to speed up processing.

Statistics gathering. Throughout the session, statistics are gathered by querying the different libraries: DI-Guy calls can be used to extract the position of the different characters or the configuration of their joints. The OBJECT MANAGER can be used to query the position of objects and their visibility from the point of view of the different characters. In addition, the internal states of all characters' simulated nervous systems are perfectly known. This data or arbitrary subsets of it can easily be recorded on a frame by frame basis for later analysis. These statistics are useful for analyzing long-term runs, and allow to evaluate whether the desired behavior is being achieved and at what rate. We point out that every simulation is perfectly reproducible and can be re-run if additional statistics need to be collected.

3. A First Example: Gaze Following

The motivation for constructing the platform was to facilitate the development of embodied models of cognitive and social development. To illustrate how the platform can be used through a concrete example, we will outline how we are currently developing an embodied model of the emergence of gaze following [5]. Gaze following is the capacity to redirect visual attention to a target when it is the object of someone else's attention. Gaze following does not occur at birth, but instead develops during a child's first 18 months

of life.

The model we are developing is aimed at testing and refining the *basic set hypothesis* [8], which states that the following conditions are sufficient for gaze following to develop in infants: a) a reward-driven general purpose learning mechanism, b) a structured environment where the caregiver often looks at objects or events that the infant will find rewarding to look at, c) innate or early defined preferences that result in the infant finding the caregiver's face pleasant to look at, and d) a habituation mechanism that causes visual reward to decay over time while looking at an object and to be restored when attention is directed to a different object. Recently, Carlson and Triesch [4] demonstrated with a very abstract and simplified computational model, how the basic set may lead to the emergence of gaze following and how plausible alterations of model parameters lead to deficits in gaze following reminiscent of developmental disorders such as autism or Williams syndrome.

In our current work, we want to investigate if the basic set hypothesis still holds for a more realistic situation, where learning takes place in a complex naturalistic environment. The platform is configured for an experimental setup consisting of a living room with furniture and a toy, all of them instantiations of the OBJECT class and built from 3D Studio Max objects. Two instantiations of the PERSON class are created, one for the caregiver and one for the baby. The caregiver and learning infant are placed facing each other. The caregiver instantiates a BRAIN object controlling its behavior. A single toy periodically changes location within a meter of the infant, and its position is fed to the caregiver's BRAIN. In a first version of the model, the caregiver's BRAIN will simply cause the character to look at the position of the interesting toy with fairly high probability (75%). No visual system is given to the caregiver.

The baby instantiates a VISUAL SYSTEM object that models a simple infant vision system. In particular, it evaluates the *saliency* of different portions of the visual field [9], it recognizes the caregiver's head, and it discriminates different head poses of the caregiver. Saliency computation is based on six different features, each habituating individually according to Stanley's model of habituation [13]. The feature maps (see Figure 1) are: red, green, blue and yellow color features based on a color opponency scheme [12], a contrast feature that acts as an edge detector by giving a high saliency to locations in the image where the intensity gradient is high, and finally a face detector feature that assigns a high saliency to the region of the caregiver's face, which is localized through orace vision. The saliency of the face can be varied depending on the pose of the caregiver's face with respect to the infant (infant sees frontal view vs. profile view of the caregiver). A similar scheme for visual saliency computation has been used by Breazeal [2] for a non-developing model of gaze following, using skin tone, color, and motion features.

Image Scale	Vision	Map Display	Animation
80×60	0.0226	0.0073	0.0476
160×120	0.0539	0.0092	0.0431
240×180	0.0980	0.0121	0.0522
320×240	0.1507	0.0113	0.0422
400×300	0.2257	0.0208	0.0507
480×360	0.3025	0.0276	0.0539

Table 1. Simulation times (sec.)

The infant's BRAIN object consists of a two-agent reinforcement learning system similar to that used in [4]. The first agent learns to decide when to simply look at the point of highest saliency (reflexive gaze shift) or whether to execute a planned gaze shift. The second agent learns to generate planned gaze shifts based on the caregiver's head pose. The infant should learn to direct gaze to the caregiver to maximize visual reward, and habituation will cause him/her to look elsewhere before looking back to the caregiver. With time, the infant learns to follow the caregiver's line of regard, which increases the infant's chance of seeing the interesting toy. However, the caregiver's gaze does not directly index the position of the object, but instead only specifies a direction with respect to the caregiver but not the distance from the caregiver. One goal of the current model is to better understand such spatial ambiguities and how infants learn to overcome them [11].

3.1. Platform Performance

To illustrate the performance of the platform given our current hardware, we made a number of measurements to establish the computational bottlenecks for this specific model. The time spent for each frame was divided into three separate measures for analysis: the time to calculate the feature maps (Vision), the time to display them (Map Display), and the time for the DI-Guy environment to calculate the next character positions and display them (Animation). Table 1 shows how the times vary with the resolution of the infant's vision system. As can be seen, most time is spent on simulating the infant's visual processing. Real time performance is achievable if the image resolution is not set too high.

4. Discussion

The platform presented here is particularly useful for modeling the development of *embodied* cognitive skills. In the case of the emergence of gaze following discussed above, it is suitable because the skill is about the inference

Property	Robotic Model	Virtual Model
physics	real	simplified or ignored
agent body	difficult to create	much easier to simulate
motor control	full motor control problem	substantially simplified
visual environment	realistic	simplified computer graphics
visual processing	full vision problem	can be simplified through oracle vision
social environment	real humans	real humans or simulated agents
real time requirements	yes	no, simulation can be slowed down or sped up
data collection	difficult	perfect knowledge of system state
reproducibility of experiments	difficult	perfect
ease-of-use	very difficult	easy
development costs	extremely high	very modest

Table 2. Robotic vs. virtual models of infant cognitive development.

of mental states from bodily configurations, such as head and eye position, which are realistically simulated in our platform.

4.1. Virtual vs. Robotic Models

Recently, there has been a surge of interest in building robotic models of cognitive development. Compared to the virtual modeling platform presented here, there are a number of important advantages and serious disadvantages of robotic models that we will discuss in the following. A summary of this discussion is given in Table 2.

Physics. The virtual simulation is only an approximation of real-world physics. The movements of the characters do not necessarily obey physical laws but are merely animated to "look realistic." For the inanimate objects, we currently do not simulate any physics at all. In a robotic model, the physics are real, of course. The justification of neglecting physics in the virtual model is that the cognitive skills we are most interested in are fairly high-level skills, i.e., we simply do not want to study behavior at the level of muscle activations, joint torques, and frictional forces, but at the level of primitive actions such as gaze shifts, reaches, etc., and their coordination into useful behaviors.

Agent body. In the virtual modeling platform, we can choose from a set of existing bodies for the agents. These bodies have a high number of degrees of freedom, comparable to that of the most advanced humanoid robots. Further, since physics is not an issue, we are not restricted by current limitations in robotic actuator technology. Our characters will readily run, crawl, and do many other things.

Motor control. Our interface to the agents in the model allows us to specify high-level commands (walk here, reach for that point, look at this object). The underlying motor control problems do not have to be addressed. In contrast, for a robotic model the full motor control problem needs to be solved, which represents a major challenge. Clearly, the platform should not be used to study the specifics of human motor control but it makes it much easier to focus on higher level skills. At the same time, perfect control over individual joint angles is possible, if desired.

Visual environment. The simulated computer graphics environment is of course vastly simpler than images taken by a robot in a real environment. For example, shadows and reflections are not rendered accurately, and the virtual characters are only coarse approximations of human appearance. Clearly, again, such a modeling platform should not be used to, say, study the specifics of human object recognition under lighting changes. The skills we are most interested in, however, use object recognition as a basic building block (e.g., the ability to distinguish different head poses of the caregiver with a certain accuracy). We believe that the details of the underlying mechanism are not crucial as long as the level of competence is accurately captured by the model.

Visual processing. In the virtual modeling platform we can vastly simplify perceptual processes through the use of oracle vision. In a robotic model, this is not possible and the perceptual capabilities required for some higher level cognitive skills may simply not have been achieved by contemporary computer vision methods.

Social environment. A robotic model can interact with a real social environment, i.e., one composed of real human beings. In our virtual modeling platform we could achieve this to some extent by using standard Virtual Reality interfaces such as head mounted displays in conjunction with motion tracking devices. In such a setup a real person would control a virtual person in the simulation, seeing what the virtual person is seeing through the head mounted display. However, the ability to experiment with vastly simplified agents as the social environment allows us to systematically study what aspects of the social environment, i.e., which be-

haviors of caregivers, are really crucial for the development of specific social skills [16]. This degree of control over the social environment cannot be achieved with human subjects. Also, the social agents may be programmed to exhibit behavior that replicates important statistics of caregiver behavior observed in real infant caregiver interactions. For example, Deák et al. are collecting such statistics from videos of infant-caregiver dyad interactions [6]. We are planning on developing caregiver models that closely replicate the observed behaviors.

Real time requirements. A robotic model must be able to operate in real time. This severely limits the complexity of the model. Perceptual processes in particular are notoriously time consuming to simulate. In the virtual model, we are not restricted to simulating in real time. Simulations may be slowed down or sped up arbitrarily. In addition, the availability of oracle vision allows to save precious computational resources.

Data collection. In the virtual model it is trivial to record data about every smallest detail of the model at any time. This is much harder to achieve in a robotic model interacting with real human caregivers. In particular, the exact behavior of the caregiver is inherently difficult to capture. Useful information about the caregiver behavior can be recovered by manually coding video records of the experiment, but this information is not available at the time of the experiment.

Reproducibility of experiments. Along similar lines, the virtual modeling platform allows perfect reproducibility of experiments. Every last pixel of the visual input to the learning agent can be recreated with fidelity. This is simply impossible in a robotic model.

Ease-of-use. Not having to deal with robotic hardware shortens development times, reduces maintenance efforts to a minimum, and makes it much easier to exchange model components with other researchers. Also, recreating the specific setup of a real-world behavioral experiment, only requires changing a configuration file specifying where walls and objects are, rather than prompting a renovation.

Development costs. Finally, robotic models are much more expensive. Most of the software components used in our platform (Linux OS, SGI OpenGL Performer, Intel OpenCV) are freely available to researchers. The lion share of the costs is the price of the BDI DI-Guy software.

All these benefits may make a virtual model the methodology of choice. Even if a robotic model is ultimately desirable, a virtual model may be used for rapid proto-typing. We see the use of virtual and robotic models as complementary. In fact, we are pursuing both methodologies at the same time in our lab [10].

4.2. Possible Extensions

There are several extensions to our platform that may be worth pursuing. First, we have only considered monocular vision. It is easy to incorporate binocular vision by simply placing two virtual cameras side by side inside a character's head. Foveation could also be added to the characters' vision systems. Second, in order to model language acquisition, a simulation of vocal systems and auditory systems of the characters could be added. Even in the context of non-verbal communication, a caregiver turning his head to identify the source of a noise may be a powerful training stimulus for the developing infant. Third, the platform is not restricted to modeling human development, but could be extended to model, say, the development of cognitive skills in a variety of non-human primates. To this end the appropriate graphical characters and their atomic behaviors would have to be designed. Fourth, on the technical side, it may be worth investigating in how far the simulation could be parallelized to run on a cluster of computers.

4.3. Conclusion

In conclusion, we have proposed a research platform for creating embodied virtual models of cognitive development. We have outlined how the platform may be used to model the emergence of gaze following in naturalistic infant-caregiver interactions. The virtual modeling platform has a number of important advantages compared to robotic modeling approaches. The relative benefits of virtual models over robotic models on the one hand or more abstract computational models on the other hand need to be evaluated on a case-by-case basis.

5. Acknowledgments

This work is part of the MESA project at UCSD. We acknowledge funding from the UC Davis MIND Institute[5] and the National Alliance for Autism Research[6]. Tone Milazzo developed the first version of the platform. Tone Milazzo and Jochen Triesch developed the initial object-oriented programming scheme. The following people contributed with valuable input: Gedeon Deák, Javier Movellan, Hyundo Kim, Boris Lau, and Christof Teuscher.

References

[1] M. Asada, K. F. MacDorman, H. Ishiguro, and Y. Kuniyoshi. Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous Systems*, 37:185–193, 2001.

[5] `http://www.ucdmc.ucdavis.edu/mindinstitute/`
[6] `http://www.naar.org`

[2] C. Breazeal. *Designing Sociable Robots*. MIT Press, Cambridge, MA, USA, 2002.

[3] R. A. Brooks, C. Breazeal, R. Irie, C. C. Kemp, M. Marjanovic, B. Scassellatti, and M. M. Williamson. Alternative essences of intelligence. In *Proc. of the American Association of Artificial Intelligence*, pages 961–968, 1998.

[4] E. Carlson and J. Triesch. A computational model of the emergence of gaze following. In H. Bowman and C. Labiouse, editors, *Connectionist Models of Cognition and Perception II: Proceedings of the Eighth Neural Computation and Psychology Workshop, University of Kent, UK, 28–30 August 2003*, volume 15, pages 105–114. World Scientific, 2003.

[5] G. O. Deák, R. Flom, and A. D. Pick. Perceptual and motivational factors affecting joint visual attention in 12- and 18-month-olds. *Developmental Psychology*, 36:511–523, 2000.

[6] G. O. Deák, Y. Wakabayashi, and H. Jasso. Attention sharing in human infants from 5 to 10 months of age in naturalistic conditions. In *Proc. of the 3rd International Conference on Development and Learning (ICDL'04)*, La Jolla, California, USA, 2004.

[7] J. L. Elman, E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett. *Rethinking Innateness*. MIT Press, Cambridge, MA, USA, 1996.

[8] I. Fasel, G. O. Deák, J. Triesch, and J. Movellan. Combining embodied models and empirical research for understanding the development of shared attention. In *Proc. of the 2nd International Conference on Development and Learning (ICDL'02)*, pages 21–27, Los Alamitos, California, USA, 2002.

[9] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40(10-12):1489–1506, 2000.

[10] H. Kim, G. York, G. Burton, E. Murphy-Chutorian, and J. Triesch. Design of an antropomorphic robot head for studying autonomous mental development and learning. In *Proc. of the IEEE 2004 International Conference on Robotics and Automation (ICRA 2004)*, New Orleans, LA, USA, 2004.

[11] B. Lau and J. Triesch. Learning gaze following in space: a computational model. Proceedings of the Third International Conference on Development and Learning (ICDL'04), La Jolla, California, October 20–22, 2004. (This volume.).

[12] T. W. Lee, T. Wachtler, and T. J. Sejnowski. Color opponency is an efficient representation of spectral properties in natural scenes. *Vision Research*, 42(17):2095–2103, 2002.

[13] J. C. Stanley. A computer simulation of a model of habituation. *Nature*, 261:146–148, 1976.

[14] R. S. Sutton and A. G. Barto. *Reinforcement Learning: an Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[15] D. Terzopoulos, X. Tu, and R. Grezeszczuk. Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, 1(4):327–351, 1994.

[16] C. Teuscher and J. Triesch. To care or not to care: Analyzing the caregiver in a computational gaze following framework. Proceedings of the Third International Conference on Development and Learning (ICDL'04), La Jolla, California, October 20–22, 2004. (This volume.).

[17] E. Thelen and L. Smith. *A Dynamic Systems Approach To the Development of Cognition and Action*. MIT Press, Cambridge, MA, USA, 1994.

[18] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291:599–600, 2001.