

Synthetic Computer Vision for Autonomous Agents in Distributed Partitioned Environments

Fis. Saskia de Winter

Isaac Rudomín Goldberg PhD.
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México

Abstract

This article describes research on the implementation of synthetic vision for autonomous agents that exist in a simulated distributed three-dimensional world that has been partitioned into sections, each of the sections being managed by a different process or computer. The agents realistically simulate the appearance and behaviour of fish.

Keywords:

Simulated Vision, Autonomous Agents, Regions, Multiuser, Distributed Systems, Simulated Physics

1. Introduction

Computer vision is usually presented as part of a robotics project. In fact, the need to formally study computer vision comes from the practical application of producing robots with the capability of seeing. Once the robots have this skill, the next natural step is to allow the robot to determine a new path or a new movement depending on the information obtained. The robot arm has to determine whether to rotate to the left or to the right depending on how the nuts and bolts have to be positioned. In a sense, this project is similar. Autonomous agents can obtain information through different learning procedures, and vision is one of them. Color, form and movement can be obtained to determine if what is seen is food, another fish from the school or a shark. Then, with this knowledge, the fish can decide to come closer or to escape. And if it doesn't swim fast enough, it might be part of tonight's sushi meal...

2. Related Work

The ACM[†] Special Interest Group in Computer Graphics has an annual conference known as Siggraph. During this week, scientists from all over the world present work related to computer graphics, three of which are more related to this paper. Go Fish! was presented in 1993 by Demetri Terzopoulos, Xiaoyuan Tu and E. Fiume from the Computer Science Department, University of Toronto. They showed for the first time a hydrodynamic movement simulation of fish based on physics. The fish have realistic motion based on a particle system which reacts to physics. Later on, in May 1996, they published another article in Artificial Life V: Proceedings of the Fifth International Conference on the Synthesis and Simulation of Living Systems, in Nara, Japan. They explain how they expanded their previous work implementing active vision algorithms for foveation and detection of interesting targets, and pursuing moving targets, among other developments. The project also included a learning process for difficult motor skills, like looping on the air for dolphins.

The ALIVE System was first exhibited at Siggraph '95. This virtual environment from Pattie Maes, Trevor Darrel, Bruce Blumberg and Alex Pentland

[†] Association for Computing Machinery

from MIT Media Lab consists of a wireless, full-body interaction with autonomous agents. The system uses a video camera to collect information from the user which is concurrently composited with the computer graphics scene. Each autonomous agent has a set of virtual sensors; which, in this case, is a set of rays shot out in a 2D plane to obtain intersections. This is a fundamental idea used in computer vision.

Artificial Dolphins was presented at Siggraph '96 at New Orleans. This virtual reality environment allows human communication with artificial dolphins using two cameras to calculate the 3D position of the user's light pen. Each different motion is a specific order for the dolphin to perform tricks. Special animation effects like foam and splashes add realism. This project was presented as a Hitachi development from the Central Research Laboratory by Norihiro Munemasa, Tsuneya Kurihara, et al. .

There is another project related to this article, which has not been presented at Siggraph because it is brand new. This is part of the graphics demos included in the new Silicon Graphics' workstation called O2. This new demo by Peter Broadwell is called gold, referring to goldfish, and shows a fish tank simulation. The user can control different parameters such as hunger, lust, and even the probability of a hatching egg to be female or male. The fish react to their mates depending on a thrill variable, but there is no real vision system implemented.

3. Description of the Problem

One reason for the development of this project is the explosion of the WWW and Internet. Every day new interactive and distributed systems are placed on Web Pages. There are new users who don't know how to program. There is a clear need for systems to be simpler, more interactive, and with more simultaneous users. Any application reaching these goals will find larger user groups. Because of this reason, this work uses distribution and interaction to allow more concurrent users to use the system. The application of computer vision in a distributed system tries to represent nature in a more efficient way. The internal calculations every viewing creature has to perform in order to understand his world is a distributed process. Fish have the viewing skills, and just like real animals, they also make decisions depending on what they observe. The system will also include a real camera to obtain certain hand positions from the user. These specific hand movements will be the sign language to communicate with the autonomous agents in their virtual world, in this case, with the fish. The four main ideas of this project are:

1. Vision
2. Distribution
3. Cooperation
4. Interaction

In other words, the main idea is to build a distributed system with autonomous agents. These agents have perception through which they obtain the relevant information to make decisions.

These ideas can be applied in many ways, and the work here presented can have specific uses in fields like production, manufacturing, design, and even entertainment, always considering the interaction and collaboration of distributed users. For example, a similar system can be implemented for a manufacturing cell in which the users use a special sign language to make an endless band roll or stop. Or a "Doom" like game can be developed where the players are in a distributed system and use another special sign language. Another application can be the software development for plastic injection where the simultaneous cooperation and interaction of various engineers can be crucial to the cast design for innovative products.

4. Background

- *Natural vision in fish* There are more than 26,000 fish species that live in any of the aqueous worlds in earth. There are so many photic environments as fish with illumination levels and spectral ranges. The characteristics of water affect the way fish see, like the illumination intensity, spectral quality, depth, etc. Even the quality of the surface will determine how much light will penetrate down below: more light will enter through calm waters than through a rough ocean. This obviously affects not only the perception sensor from fish but also the way they are colored. They will usually be darker on the dorsal part and lighter on the ventral part so that they appear uniform underwater. There are some fish which can only find their prey by encountering the silhouette, which is a similar process used in this work with the real camera. The fish have an adaptive behaviour during the day. For brighter light, the cones are concentrated on the front part while rods are concentrated for dimmer light[‡].

Fish have a spherical lense which moves within the globe to focus the image and to remain constant for underwater vision. That is, elements like the lens and the cones and rods move constantly during the day for best performance in their perception mechanism.

- *Active Vision*

[‡] This can be simulated by using several cameras with different resolutions instead of just one for each eye.

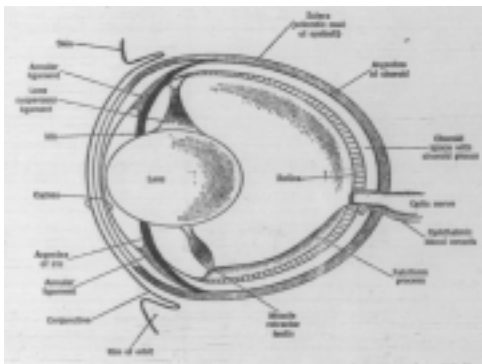


Figure 1: Sectional diagram

Active vision systems can physically control parameters such as position, orientation, focus, zoom, etc. These parameters can be controlled through mechanisms which depend on external stimuli, and on the tasks to be reached. Active vision has four main elements: attention, space selective perception, resolution and time. Active vision research concentrates more on the fields of attention, fovea perception, gaze control, hand-eye coordination and robot architecture integration. Active vision should be used on systems which have the need to identify dynamically and automatically the changing pre-requisites of a vision task. These pre-requisites can be manipulated through the vision system to stabilize them.

- *Vision*

Computer vision based systems derive from robot vision investigations. The following is a set of important concepts used in this field:

Registration

Registration is an alignment of different or equivalent data. There are 3 main techniques:

1. Manual techniques using color and movement: Two images are aligned together to obtain differences when motion is pursued.
2. Semi-automatic techniques based on sections: The images are marked down and then referenced later with a tracing mechanism.
3. Optical flux: A set of vectors between pixels helps to estimate the local movement. Error is minimized.

Segmentation

Segmentation is the process to obtain significant regions from images. There are 4 methods to do segmentation:

1. Statistical classification method: Image blocks are compared within a specific threshold value to ob-

tain a staired or jagged image. This is also known as aliasing.

2. Region-growing algorithm: A seed starts to grow when it encounters neighbors which are part of the same region. Another way is to analyze large regions to subdivide them when elements with different intensities, color or texture are found.
3. Boundary-based segmentation: Edge detectors in 3D are used. There are 3 kinds of edge detectors: Surface patch adjustment, surface-tracking algorithms and 3D operator application as gradient shadowing.
4. 'Snakes' or Interactive deformable contours: The user draws an approximate contour of the area to be analyzed, and then forces are applied in a dynamic simulation to find the real edges.

Edge detection

Edge detection is a very complex process. There are specific factors which can complicate the calculus. For example, a shadow or a reflectance change can be considered as an edge. Segement discontinuity should also be discarded because of image noise. And even textures on objects can also detect false contours. A contour on some direction represents a notorious variation on the intensity function perpendicular to the edge. Then, to caculate an edge detection is just a local extreme computation.

Stereoscopy

Most living creatures have a binocular perception system. Fish are no exception, although there are many species which don't have this feature. Considering those who do, to simulate a binocular perception system it is necessary to create two images which observe the same object: one for the left eye, and one for the right eye. To match the same point observed from the two images is the main task for a stereovision algorithm: this is known as the correspondence problem: Which point from the first image matches a specific point from the second image? Upon this question, some other are necessary: What happens if there is a series of points that, due to eye positions, are aligned? What if the object is transparent? Because of these and many other questions, the correspondce problem is considered an ambiguous dilemma. Fortunately, there exists three kinds of restrictions which diminish the uncertainty of stereovision:

1. **Geometric restrictions generated by images:** An epipolar axis connects the optic centers of the images. The optic centers are at focal distance f from the corresponding image. Both optic centers are connected by the epipolar axis. The epipolar axis also intersects the images at the epipolar points. The epipolar point of one image is connected to the point to be matched on the other

image. Thus, the area to look into has been reduced from a 2D area to a line: the points that can match are the ones on the line: precisely the line which connects the epipolar point with the point to be matched. This is also known as the epipolar constraint. See figure 2.

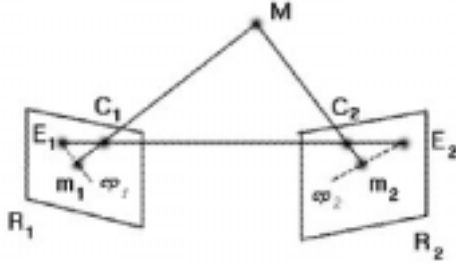


Figure 2: Epipolar constraint

2. **Geometric restrictions generated by observed objects:** The distance between the object and the vision system may vary on almost at any part.
3. **Physical restrictions:** These are due mainly to the way the light interacts with the object.

Once the two possibly matching points are detected, they are verified by matching reflexion values, or by matching a group of pixels or regions. But then we have to consider that the reflection values may vary between the two images due to light positioning: light will be reflected differently for the same point. This is another reason to use Lambertian models instead of Blinn or reflective ones, even though in real life there are just very few matte objects.

- *3D coordinate computing from 2D images*

Each fish must create two 2D images from the surrounding world. This is much like a set of cameras, one on each eye. Each eye or camera observes the same point, and the computations are followed for the single point M:

m_1 and m_2 are the pixel coordinates from each image. f is known from the cameras used on the specific graphics library; for example, in Open Inventor the class SoCamera includes the field focalDistance which is an SoSFFloat type. Cameras can also be used in VRML, thus with Java. Orientation and fieldOfView are two examples of the Viewpoint Node in VRML. The optical centers are calculated from the camera's calibration. As C_1 and C_2 are known, d_{12} can be obtained. Then, the coordinates (x,y,z) from the point M are obtained through:

$$d = v_2 - v_1$$

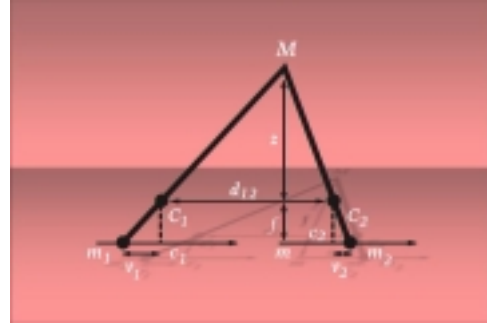


Figure 3: Camera diagram

$$x = \frac{d_{12}(v_1 + v_2)}{2d}$$

$$y = \frac{u_1 + u_2}{2}$$

$$z = \frac{d_{12}f}{d^2}$$

This is the way to calculate the 3D coordinates, and by connecting lines, the complete 3D object can be rebuilt. A fish form can be recognized, and a decision can be taken, based on the shape of the fish. Shapes, colors and textures can help define the behaviour to follow, like coming closer or escaping. Two fish can be similarly shaped, but with different texture.

Object recognition can be a complicated task because the objects are not always seen from a front position, sometimes a fish might be observed from a 3/4 position. The best way to approach this problem is to have three cameras instead of two, but this research work is a simulation of a vision system, which implies using only two cameras. Then, careful procedures have to be followed to recognize the different fish while considering the various viewpoints.

5. Our Approach: The Vision system in a partitioned world

Vision systems in sectioned worlds pose particular problems. In the following we discuss the process and some of the decisions taken:

World subdivision The world is subdivided in regions. In each region, there are fish and users. For more fish and users, more regions are needed. There exists a multi-server which calculates each of the animation positions, per region. Also, each of the workstations has a copy of the region where the user lives. Then,

both the region multi-server and each workstation calculate with different algorithms the new positions. If the positions computed differ from a specific quantity, then the server wins and has to send the corrected information to the workstation. This process is followed to avoid an excessive message traffic in the network. If all the multi-server's calculations are sent, the system would be clogged and very slow. Only at certain time periods, the information is verified, but each workstation should always have the correct information. The equations are to be solved either by Runge-Kutta method or by Euler method.

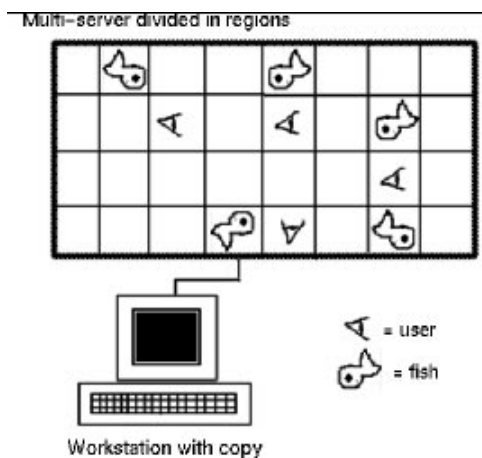


Figure 4: World subdivision

Data Collection

Data collection for this project consists on generating 2D images which show exactly what the fish is observing. There are several ways to do this. In OpenInventor, one could use the available offscreen renderer, or even generate other views in other OpenInventor windows. We have, for the moment, chosen to do it differently, by shooting rays from each eye within a 300 degree angle. When the ray intersects with an object, that is when the ray finds a pixel with a different color from that of blue (water), then, the two images should be rendered. These images have to go through a recognition process to understand every element on the image. This rendering process is also known as stereoscopy, which is similar to the natural human vision.

In our partitioned environment, each fish can be considered to live in a region, (a cube). Whenever the fish has to see outside its region, it must request information from neighboring regions. It is equivalent to having pre-rendered images of projected worlds on each cube face, and use them as background for the rendering of the other objects within the region.

Most likely, only two of the six faces will be rendered and analyzed, unless the viewing angle converts more than two faces, which then means rendering three faces, this depends on the fish's viewpoint. Now, the fish tank is divided into regions. The image rendered on the cube's face is the accumulation on the images from the cube's faces from the neighbors. Depending on the viewing range of each fish, there will be more or less images accumulated.

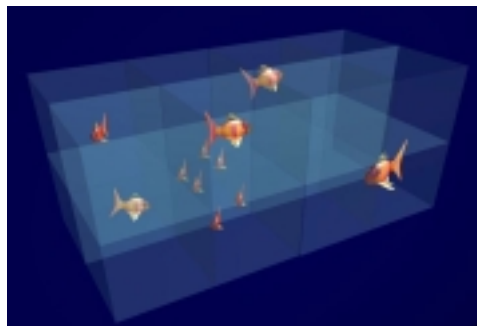


Figure 5: Regions and accumulation of images

Once the fish has rendered the left and the right eye, the images should be analyzed to yield the significant points and coordinates. There are many segmentation procedures and it is necessary to test several algorithmic applications to determine the fastest and most effective algorithm. 64 x 64 pixel images will be analyzed, although image size can be varied. This size is chosen because of rendering time considerations.

The user can also be part of the fishtank. Any user can live at one region at a time, and the user can move from region to region to observe the reactions of the fish. Both fish and users move from region to region. In fact, each region has a list of fish inside and a list of users inside. The main difference between the fish and the virtual user is that the fish have an implemented mechanism to perceive their world, and the user will take advantage of the graphics library camera; that is, the user can already see, and has no need to use edge detection for image analysis.

Edge detection

Besides analyzing the fish's renderings, edge detection will be used to create the user's silhouette. The user will be positioned in front of a physical camera which will be the communications interface with the fish tank. The user will interact with the virtual fish by using hand movements, and every position will mean a different order, for example to come closer or to flip. Edge detection will determine if the right hand was raised, or if both hands were raised. This is the similar procedure used to analyze whatever the fish is

observing, only with different type of images. That is, the edge detection procedure is used for the autonomous agents, or for the images generated from the hand movements from the real user.

Stereoscopy

A stereoscopic mechanism is used for the 300 degree view angle for each fish. As explained above, 3D object coordinates are obtained to rebuild any object on the scene to specify which object it is.

Distribution

The initial project was started on Open Inventor as the graphics library on the Unix platform. For the initial proposal, PVM was considered as the communications language between workstations. Data consistency is an important factor: any user on any workstation has to observe the same fish tank, although each user might only observe a smaller region, or might observe the fish tank from a different point of view (from the backside, for example). The project will also use VRML and Java. VRML will allow the construction of a friendly user interface plus the usage of 3D. Java will create all the necessary elements for the interface such as buttons and a multi-threaded environment. Each fish will live in a different region, and there might be more than one fish per region. The same is for users: each user might be looking at a specific region, and the user can move from region to region. In other words, as has been already mentioned, the fish tank is divided into regions which are controlled by a multi-server (a server per region). Each agent (fish or user) calculates its own perception: whatever he observes in the regions where he's currently at.

In short, the initial goals are:

1. Autonomous agents with behaviour will be implemented. Each agent is a fish and its motion control is based on the decisions it will take according to the information collected.
2. A vision system for each fish will be implemented. This will help analyze the surrounding world to collect information and then make the appropriate movement. The vision system consists of rendering at least two images for each fish when an object in front is detected through ray-shooting. Each image will be analyzed to obtain a list of matching 2D points to obtain the 3D coordinates of the objects observed. Through this listings, the 3D objects observed can be rebuild. The fish will decide how to behave afterwards.
3. Virtual users can also move between regions. These users observe how the fish react, but do not analyze rendered images.
4. A physical camera will be installed and the adequate software will be developed to recognize the

user's hand movements. This will be the communication language with the fish, which also have to 'see' what the user is doing.

5. All the previous steps should be implemented in a distributed world. This means that the fish tank is a multi-server with regions which communicate. A region can tell its neighbor when a fish will enter this new region. Each region knows how many fish and how many users are there.

6. Current Status

This project is in progress. The following work has been completed:

Open Inventor was used to create the first trials on the ray-shooting and the rendering of the scenes. The fish have simple movements like translation and rotation. The current number of fish is between five and seven, and for each new fish in the tank, the system becomes slower. This happens because it is still not a distributed system, and all the computing is done on the same CPU (Silicon Graphics Indigo Extreme). Once threads are implemented, each workstation will be in charge of some amount of computations so the simulation will run faster.

The computer vision implementation will have two parts:

1. The fish need to have a perception system to observe the fish tank in which they live. According to the information collected, the fish 'understands' whatever it is looking at, and the fish reacts like an autonomous agent and makes a decision. For example, the fish has to distinguish between a predator or food. If the system is effective and the fish 'sees' fast enough, then it can take the decision fast enough to escape from a shark. Also, there exist virtual users which navigate between region and region to observe the fishtank.
2. The system will also have a physical camera as a peripheral connected to the workstation. This camera captures the image of the user interacting with the fish tank. This way, there is also data collection through the physical camera and this is part of the fish's surrounding world, as if the user were another fish. For example, if the user raises his left hand, that might mean for the fish to come closer, or if the user raises his right hand that might be for the fish to go away, just like some fish do in real life.

The perception system has its origin on the eyes of each fish. Every animal watches from its own viewpoint, and every fish collects different information of the world. 2D rendered images will be the result of everything each fish is observing. It is just like two instantaneous cameras on each fish's eyes. The photos

will provide the information on how to behave. Each 2D image is a collection of the 2D images of the parallel regions; that is, they accumulate up to certain distance. For example, a fish is looking staring forward towards a region in a 0 degree angle. Then, this fish will see directly what there is in front, and this will be the sum of the parallel walls from the neighbor regions. See figure 6.

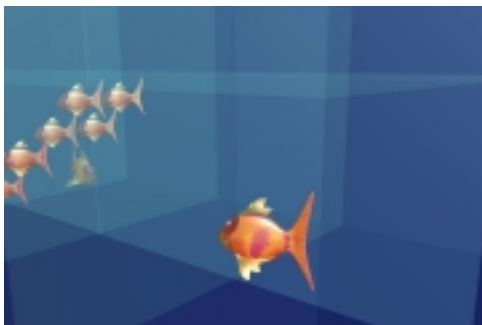


Figure 6: *How the fish live in the regions*

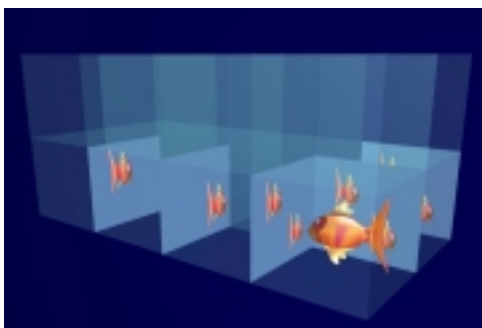


Figure 7: *How the other walls are projected*

The implementation already has a rayshooter on Open Inventor. From a reference point, each ray is shot, and it specifies whether it has hit an object or not. After detecting the existence of an object, the image should be rendered to be analyzed with edge detection.

Open Inventor is an object-oriented language which created a hierarchical tree for each scene. The ray can determine which object was shot by following the hierarchical tree. For example, if the scene has a red fish and a blue fish, and only the red fish was intersected, then, while following the tree it will be determined that the red fish was hit. But this is not the way a vision system works. Humans don't generate a hierarchical tree for everything observed. Any living creature collects and analyzes information through a sensorial system (cones and rods, brains), but never through

a description process or through messages. The same procedure should be followed on this study. No hierarchical trees or messages are allowed to analyze the elements on the scene.

The next step is to image analysis; that is, to calculate the 3D coordinates of every object observed on each eye.

For the physical camera, an edge detection system is already implemented in "C". There are 5 methods which analyze data. Each colored image is translated into a greyscale map. These methods are:

1. Vertical edges method: A vertical grey level average per pixel is calculated (y). A binary map is obtained after the average is compared to a threshold value.
2. Roberts method: The rate of intensity changes at 45 degree and 135 degree is calculated.
3. Absolute value method: Absolute values on intensity averages are computed.
4. Sobel filter: Calculates the horizontal components of a smooth gradient.
5. Prewitt filter: Calculates the vertical components of a smooth gradient

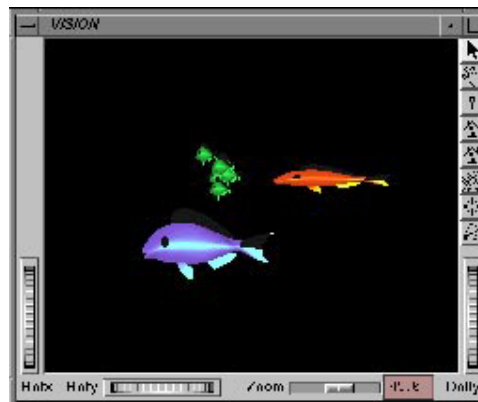


Figure 8: *System image*



Figure 9: *Left eye*

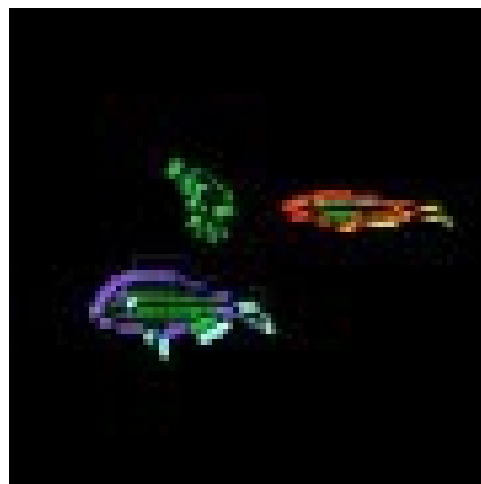


Figure 11: *Edge detected image*



Figure 10: *Right eye*

Once the images are edge detected, the autonomous agents will determine whether to move forward, backwards, left or right. These algorithms have yet to be transcribed into Java and VRML. The interface for everything the user is observing has been partially implemented in VRML and Java. This currently includes one fish observing a screen. This area has different images which change randomly: a blue cylinder, a yellow sphere, a red cone and some fish. The fish observes an image and reacts accordingly. This system's initial stage is based on colorimetry rather than on edge detection. This means that the fish still reacts exactly the same if it observes a red cube, a red cone or even a red shark. Form will determine new behaviour. These algorithms have to be written yet. Every time a user enters the system, it will be registered as a virtual user

roaming in the fishtank. Each user lives in one region, and each region has a list of all the users inside.



Figure 12: *Fish watching tv and reacting to images*

At this point, each workstation has a copy of each agent in that specific region. This region also computed the motion and perception calculations needed, and they are compared every certain amount of second with what the multiserver has calculated, as previously explained.

7. Preliminary results and Conclusions

As mentioned above, the system is partially implemented in Open Inventor. This system includes a fish school with certain simple translation and rotation movements. One of the main problems with the current system is that each fish wireframe consists of a large amount of vertex points. It is an IndexedFaceSet

from Open Inventor, and every translation and rotation takes longer if there are more fish on the scene.

So far, the rayshooting mechanism allows the scene camera to detect whether there are objects in front or not. If there exists any object, in this case fish, then the system automatically renders the left and the right eye of the scene camera, just as if the user were another fish. These images are 64 x 64 due due rendering amounts of time. This can be a very slow process, every time the camera detects an object, it renders 2 64 x 64 images, and quickly calculates the 3D coordinates of the 2D points depending on the values of the optical centers, the focal point, etc (See stereoscopy above). This again justifies the use of a distributed system: each agent will be in charge of its own rayshooting and its own calculations. Once this is implemented, it should work faster. The edge detection filters have been applied to different images, and most of the times the best filters are the Sobel and Prewitt: the amount of noise reduction and edge sharpness is more effective than those yielded by the other 3 filters applied.

The VRML and Java user interface has also been implemented. The fish has a primitive behaviour depending on the image observed. For example, if the image on the screen is a blue cylinder, the fish will move downwards. Its motion is controlled by a particle system based in physics. The next natural step is to implement efficient algorithms to detect form and shape, to be used also as a desition variable. Threads will be used for the distributed system.

8. Bibliography

- [1] Ballard Dana H., Brown Christopher. Computer Vision. Prentice-Hall. New Jersey. 1982.
- [2] Computer Vision for Computer Graphics. Course Notes for SIGGRAPH '95.
- [3] Carlbom Ingrid. Introduction to Registration and Segmentation. Digital Equipment Corporation. Cambridge Research Lab. April 27, 1994.
- [4] Faugeras Olivier. Three-Dimensional Computer Vision- A Geometric Viewpoint. Mit Press. Cambridge, Massachusetts. 1993
- [5] Fernald Rusell D. The physiology of Fishes. Chapter 6 Vision. CRC Press. Boca Raton, FL. 1993.
- [6] Fu Daniel D., Hammond Kristian J., Swain Michael J., Vision and Navigation in Man-made Environments: Looking for syrup in all the right places. Department of Computer Science. University of Chicago.
- [7] Marr David. Vision- A Computational Investigation into the Human Representation and Processing of Visual Information. W. H. Freeman and Company. New York. 1982.
- [8] Reynolds Craig W. Flocks, Herds, ans Schools: A Distributed Behavioral Model. Symbolics Graphics Division. SIGGRAPH '87.
- [9] Swain Michael J., Stricker Markus. Promising Directions in Active Vision. University of Chicago. NFS Active Vision Workshop. 1991
- [10] Terzopoulos Demetri, Tu Xiaoyuan. Perceptual Modeling for the Behavioral Animation of Fishes. Department of Computer Science, University of Toronto.
- [11] Terzopoulos Demetri, Tu Xiaoyuan, Grzeszczuk Radek. Artificial Fishes: Autonomous Locomotion, Perception, Behavior, and Learning in a simulated Physical World. Department of Computer Science, University of Toronto.
- [12] Terzopoulos Demetri, Rabie Tamer, Grzeszczuk Radek. Perception and Learning in Artificial Animals. Department of Computer Science, University of Toronto. Artificial Life V: Proceedings for the Fifth International Conference on the Synthesis and Simulation of Living Systems, Nara, Japan, May 1996.
- [13] Thalmann Nadia Magnenat, Thalmann Daniel, Renault Olivier. A Vision-based Approach to Behavioral Animation. 1990
- [14] Thalmann Nadia Magnenat, Thalmann Daniel, Renault Olivier, Noser Hansrudi. Navigation for Digital Actors based on Synthetic Vision, Memory and Learning. 1995.
- [15] Thalmann Daniel, Boulic Ronan, Noser Hansrudi. Automatic Derivation of Curved Human Walking Trajectories from Synthetic Vision. 1994.
- [16] Wernecke Josie. The Inventor Mentor. Programming Object-oriented 3D Graphics with Open Inventor, Release 2. Addison-Wesley Publishing Company. 1994