Pavel kazatsker

CPRE 585 X Project Proposal

Introduction

Self-detection and the concept of the self have been a sizeable research area in the field of developmental robotics. Research has shown that self-detection is essential functionality for tool use in intelligent biological beings. It is, therefore, necessary to examine mathematical representations of self-other separation so that self-detection can be better implemented in robots.

A necessary step for this is implementation of self-other separation and simple tool use in an artificial environment. The simplification of the environment creates the possibility for a simple task with clear-cut objectives to be added on top of the task of self-other separation.

Basic Idea

The basic idea I'd like to demonstrate is that self-detection is a major step towards solving a simple task. That is, given a very simple algorithm and some self-detection functionality, a robot can perform a seemingly sizeable task like playing a game. The only instructions explicitly give to the robot is that the objective of the game is to chase objects around the screen.

This experience as a whole is very similar to that experienced by people when they play games or even use any type of machine. When a person picks up a game, the first thing he'll do is wiggle the control or press buttons to see how the screen reacts. Furthermore, there's a common personal experience where interfaces make it hard to distinguish the part of the interface the user controls. In those cases, the software is very difficult to use effectively.

Related Work

A number of previous papers have been written on the subject of Self-Detection. Research has shown that the ability to identify the self and proprioceptive effects in the visual field are an ability that is unique to humans and great apes (Gallup, 1970). This indicates that this skill is, to some extent, necessary for a more advanced understanding of intelligence.

There have been previous discussion of the use of video games to illustrate the use of learning methodology. Generally, games are used to teach information the place of a traditional classroom setting (Shaffer, 2005). Our use for these games is somewhat different, however. Although we are using the game as a simplification of the real world in the use of learning, we are using games to teach physical coordination instead of facts from a classroom. There doesn't seem to be literature discussing the use of video games to teach coordination.

The last set of relevant papers are those dealing specifically with self-detection in robots. The most popular approach for self-detection seems is a measurement of the delay between action and reaction in the robot's perception (Gold et. al. 2004). This general idea will ideally be maintained in this project but the exact previous measurements involved with self-other separation seem incompatible with with the scenario of the games.

Equipment

With the exception of the physical robotic platform, the equipment used for this study is largely readily available and easily affordable. The robot platform is an upper torso humanoid

torso with Barrett arms and hands. Only the left arm of the robot is going to be used to control the game. The robot's hands are equipped with touch sensors and vibrotactile sensors but neither of those modalities are necessary for this study. Of the modalities that are native to the arms and hands, the only one used will be proprioception.

The robot's head is equipped with two commercially available webcams. Only one of these webcams will be used. In order to get decent visual data, the webcam should be able to take in images at at least 15 frames per second and 640 X 480 resolution.

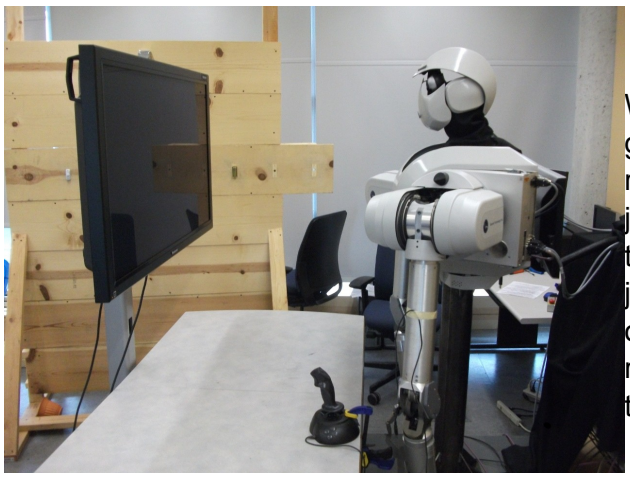
The upper torso humanoid robot is quipped with two arms and two webcams but only one of each will be used in this study.

The controller used with be a commercially available microfost sidewinder joystick

A commercially available joystick will be use to actually play the games. The joystick chosen was a microsoft sidwinder joystick. The major parameters considered were the size of the joystick and the width of the base. The size of the joystick is important as the robot lacks the ability to grasp small objects. The base has to be sufficiently large so that it can secured to the table.

The Television selected for this project is also commercially available and doesn't have any special equipment unique to this project. The only relevant parameter is that the television has to be sufficiently large to take up the majority of the webcam's visual field.

A large Television will be used to display the games.



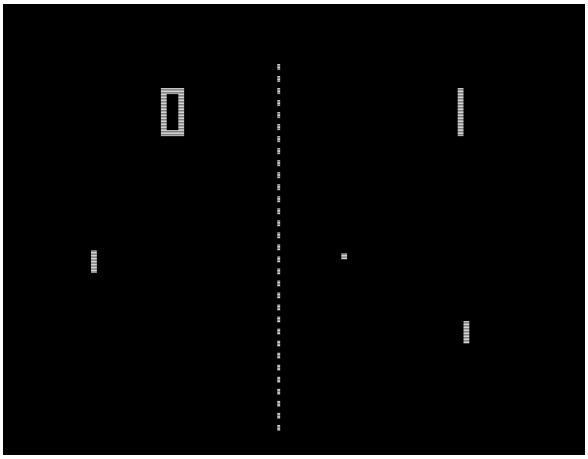
All the equipment together

When all the stuff is put together it looks a little goofy with both the television and the joystick really close to the robot. The proximity of the joystick to the robot is necessary in order to get the full functionality out of the joystick within the joint-space of the robot. Similarly, the proximity of the television is intended to get as much resolution as possible our of the combination television and webcam.

Games

This project was first introduced as teaching the robot to play Pong. This was, however, seen as too singular of a task for an entire study and was, therefore, extended to include two additional tasks.

Pong was credited as being one of the first arcade video games in the early 1970s. The game was meant as a tennis simulator. Each player controlled a dial that moved the paddle along a vertical axis on either side of the screen. The objective was to continuously deflect the ball away from your side and to keep it from getting passed the paddle. The original game was quite simple and just featured the ball bouncing between the two paddles. The trajectory of the ball after a collision with the paddle would depend on where the ball hit the paddle. The closer the collision was to the paddle's corner, the more extreme the return angle would be.



A screenshot of hte original Pong game. The game had very simple graphics and no color. A player controlled each of the paddles and a score was maintained to keep track of how many times the ball got passed your opponent.

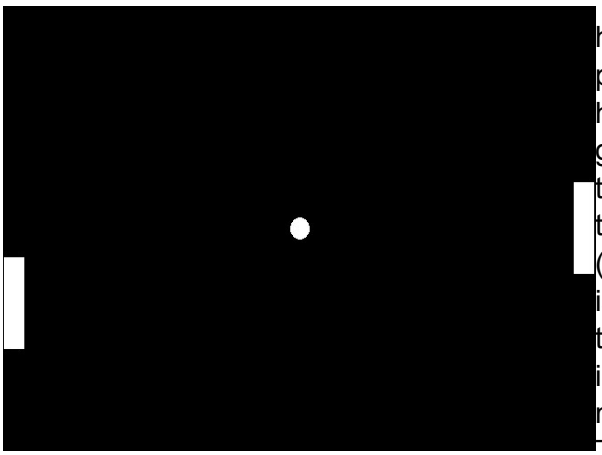
Pong eventually inspired a similar game where the paddle would, instead, move along the horizontal axis and had to deflect the ball at colored bricks that are situated opposite the paddle. The original version of this game was developed in 1976 and called breakout. The game featured virtually identical controls with a small dial that controlled the motion of the paddle on the screen. As this was later, the game looked a little bit more modern with notably higher definition textures and a number of colors. Countless breakout clones were made with better graphics, additional features, and unique themes to each game. Among the more popular clones was a game called Arkanoid that came out in the 1980s.

The classic games could not be used for a variety of reasons. The original games used a small dial to control the paddles. This was not seen as an effective mode of control as the robot could not effectively manipulate a small dial. Furthermore, randomly learning to manipulate dial would not be feasible for this task.

The games also have a number of visual elements that make it hard to use them for my purposes. The original games were very simple and had rather low resolution elements. The visual model used in this project requires that graphical elements have discernable textures and are somewhat large. Furthermore, images taken against a black background are prone to heavy reflections which further throw off the visual model.

The original game of breakout had low resolution textures but introduced a number of simple colors. This game was played single-player and the objective was to hit all the bricks on the screen without letting the ball get passed the paddle.

A new version of these games was, therefore, made to better accommodate the needs of the needs of this project. The control problem was fixed by creating versions of the game where the paddles' positions are manipulated by moving a joystick. The new versions of the games will have much larger in-game sprites with colored textures. The background in the modified games will also be white to manage the problem with the reflections in the game. Lastly, the movement of the paddles in the games is going to have to be modified away from what is traditionally experienced in those game. Many of the games have the joystick control the velocity of paddle whereas these version would control the location of the game directly.



A modern open-source version of Pong. Color was once again absent from the game but the higher resolution is evident

These two well-established games both have the rather prohibitive limitation of having the player sprite move along a single axis. This is not, however, an assumption of the system so a third game will be introduced in order to demonstrate that this system can handle a player that moves in two dimensions. This game will feature a simple (square) player sprite and a target that the player is supposed to hit using joystick controls. In order to keep the algorithm working with the same instructions, The target will be a simple object moving in some small and predictable manner. This is necessary to maintain the distinction between the object that should be identified as "other" and the background that's present in all three games.

Learning Process

The robot will have no foreknowledge of the task at each run of the process. The robot will gain a concept of itself within the game and, therefore, the knowledge of how to play the game each at each activation of this process. This is somewhat in line with findings in biological learning environments.

The first thing that happens at each activation is that the robot undergoes pre-programmed motions to graps the joystick. There is substantial research involved in grasping and learning to grasp objects but that is outside the scope of this research. The grasping will be treated as a given so that the focus of the experiments will be focused on motion and detection.

Once the robot has the joystick in hand, it will explore the functional area of the joystick. The functional area is defined as anywhere the robot can move while holding the joystick without exceeding predefined torque limits. In order to find the geometric limits of this functional area, the robot will move the arm in eight directions until the predefined torque limits are reached. The robot will then store the locations where the torque limits were reached and the convex hull surrounding all those points in considered the functional area of the joystick.

The real process begins with the robot randomly babbling around this defined functional area. At each movement, the robot will observe the screen and at this point attempt to determine what part of the screen it controls and what effect each type of movement has on the game. The number of movements required during this phase is among the results that this experiment will idea produce. The only requirement for this babbling phase is that the robot must gain all the information from this phase in a short period of time (less than fifteen minutes).

Once the robot has completed the babbling phase, the game will start. The robot will then have to use previously gained information to determine what part of the game it control and how to chase down the ball. At this point, the self-other separation will be, somewhat, tested. There will actually be an object that should be classified as other during the course of the game. At each time step, the robot will run through a subset of the motor commands memorized during the babbling phase and perform the one that most closely brings the “self” object to the “other” object.

This algorithm will be used on all the games. In two of the games, the motion of the player sprite is confined to a single axis. This is not explicitly assumed or refuted by assumptions given to the robot. The robot will be programmed to move the “self” sprite as close as possible to the “other” sprite. In those cases, the locations where the self is closest to the other is the one in which the paddle will properly block the ball.

Assumptions

There are various assumptions that are required to simplify some of the real-world complexities involved in playing these games. A number of these assumptions are introduced to restrict the scope of the study to something manageable. As mentioned before, the robot will start with the Joystick in hand. Although the thought of finding and successfully grasping the joystick is academically interesting and challenging, its simply not seen as a relevant part of this project.

The other item that was scoped out of the project was the idea of the robot learning to play the game. game play has historically been among the largest research area in artificial intelligence as a field. All concepts of scoring and game performance are, therefore removed from the robot side of the algorithm. The objective of intercepting the ball is, therefore, programmed directly into the algorithm. An alternative to this methodology was to program the physics of the game into the game playing part of the algorithm and have it accurately predict toe trajectory of the ball. This idea was ultimately rejected as involves programming actual game rules into a learning agent and would add an unnecessary complication into the system.

Other assumptions were introduced to simplify some of the individual subsystem and make them more manageable. Some of these assumptions are somewhat natural to the setup while others tend to be somewhat more counter-intuitive.

The visual model used for this project works under the assumption that only moving objects need to be tracked and that the tracked objects will move in directions that are normal to the robot's field of vision. On the one hand, this is somewhat natural as these games are intended to be played with the screen upright and with the objects moving along the plane of the screen as is customary in two-dimensional games. This does, however, serve as a substantial simplification that is assumed by the vision algorithms employed in this project. In theory, there is nothing to prevent the sprites in the game from moving in three dimensions. The apparently changing shapes and sizes of the objects would throw off the vision tracking software implemented.

The least intuitive simplification that implemented for this project is the postiion-based controls that are implemented for the games. This was done as a number of events that are likely to be identified as robot-introduced events would, in fact, be created by the nature of the game. A velocity-based control scheme would require a self-other separation scheme that registers temporal events based on the changes in velocity. Such a scheme would be thrown off by false events that are introduced by the game itself. Consider, for example, the very common situation of the paddle hitting the side of the wall and stopping. This would register as an event that would likely be expected to be generated from the robot. This misinformation would create additional difficulty in separating the self from the other in the game.

This assumption does scale back to the overall goal of self-other separation in the real world. The assumption that objects on the screen move directly with motion in the real world makes sense in the context of the real world after the television component is removed. Simply put, its more natural to assume that the things you're looking for are going to move with direct correlation with the movements imposed on them as opposed to your movements creating a change in velocity.

Another somewhat unnatural assumption is that the robot will have a babbling period where the only moving objects in the robot's visual field are those controlled directly by the robot. These types of games generally don't have game modes in which they get to practice moving around the functional area. This is, however, more consistent with the human developmental people. People have many years to develop their coordination before any form of test is imposed on their reaction.

Vision Tracking Model

The vision model alluded to in his proposal is based on a paper from the University of Massachusetts Amherst. The paper discussed partitioning objects into a visual field into the various rigid components. This same methodology will be used to extract the individual objects in the visual field of the robot when its trying to play the games.

In short, the visual model uses a functionality already present in the OpenCV the open computer vision library to find features in an image. The features are then tracked from a given frame to the next. Given the assumption that relevant motion is all normal to the vision of the robot, the features of the rigid components will should maintain a constant distance from each other. The features that maintain a constant distance are, therefore, grouped together and

maintained as unique components. These components are then measured and each maintain an identity as being self or other.

Here is the basic psuedocode that was implemented to paritition the rigid components in the visual field as well as get their delays for each movement.

```
Process Actuator (robot):
1.      time1 = time();
2.      move and wait
3.      shared_stoptime = time();
4.       shared_startime = time1;
5.      wait for command
6.      goto 1

subroutine Sensor(components);
1.      get frame
2.      if not first frame
3.              update components
4.      for each components Ci
5.              if Ci is moving
6.                      if shared_startime + START_DELAY ≤ time() ≤ shared_stop + STOP_DELAY
7.                              mark Ci self
8.                      else mark Ci other
9.              if Ci self marks / Ci total marks > .75
10.                             consider Ci as self
11.             else consider Ci  as other

Process Controller
1.      arrLoc[NUM_MOVEMENTS], arrJoint
2.      Components = {}
3.      for i = 1 to NUM_MOVEMENTS              ;babbling phase
4.              issue movement command
5.              sensor(components)
6.              arr[i] = average of “self” component locations
7.              arrJoint[i] = joint angles
8.      sensor(components)
9.      index i = indexOf(minimum distance between average “other” components and arr[index])
10.     moveto arrJoint[i];
11.     goto 8
```

Evaluation

This project will have a variety of evaluation criteria at each of the steps of the project. The first thing that will likely need to be evaluated is the performance of the visual model in light of the various circumstances introduced by this project. The visual model Is quite essential as it is a dependency of pretty much each other module in the rest of the project. Experience has shown that the most essential criteria for the visual model is its ability yo maintain consistent identities for given components over longer periods of time. At the conclusion of the project, the visual model will be evaluated given the simple measure of how long the main components (the primary ones representing the ball and the paddle) can keep their identities from the time they're

first identified. Since there is a substantial amount of noise introduced by the television as well as other factors. The existence of small, fleeting features that go away after only a few frames doesn't really matter and will not reflect poorly on the visual model.

The ultimate success of the project will be judged by the success of robot agent in its consistency when playing the various games. The main output is meant to be the ability to play Pong. There will be multiple testing criteria in the testing of the Pong game. The simplest one is a subject test against a human opponent. Player can come in and judge the robot's playing ability by its ability to keep the ball in play and, perhaps maintain a lead. The ability to win is not, however, necessary for the success of the project, and any agent that can even put up a fight will be seen as a success in the grand scheme of the project.

Other potential criteria are somewhat more objective. Counting the number of consecutive times the ball is kept in play by the robot against a perfect opponent is a telling measure of how well the robot can perform given this algorithm. The nature of the game allows for a decent amount of variability within this measure. The opponent is capable of effecting the difficulty of the game by hitting the ball at various angles. I will, therefore, create a performance measure with opponents with two different play style. The first opponent will always attempt to hit the ball back with a horizontal trajectory while the other one will hit the ball within a random location making it more difficult to follow the ball. Obviously, its expected that the robot will perform better in the situation where the opponent always hits the ball back straight.

The performance in the breakout game will be similarly evaluated. The objective is to be --as consistent as possible so the result will be evaluated by the number of consistent hits the player agent has before the agent fails to return the ball. This is somewhat counter-intuitive as the true objective of these games is to clear all the bricks. The notion of bricks is not, however, explicitly programmed into the robot or presented to the robot in any way. Bricks are, therefore, a purely incidental element that does not explicitly effect player performance.

The objective for the third game will be measured somewhat differently. Since there is no concept of hitting or missing a ball on the screen, the performance in the game will,instead be evaluated on how long it takes the player agent to reach its target at each run.

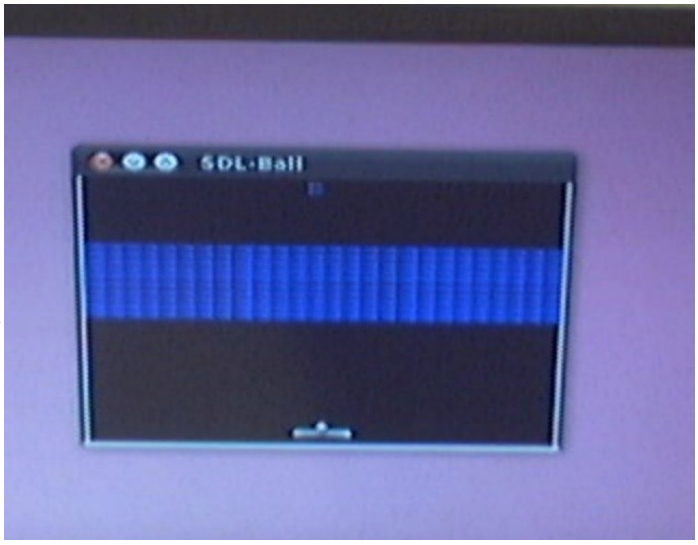
For all of these games, ther performance of the agent Is supposed to improve over time. Some measurement will be taken into account to measure how much the performance of the changes as time goes on. The perceived effect of the joystick will be updated as time goes on. If these updates are performed correctly, the expectations of the robot will likely become more accurate as time goes on. Furthermore, the notion of the “self” versus the “other” is likely to grow stronger with the progression of time. The perceived probability that the paddle is “self” and the ball is “other” should grow and converge as time goes on. This will be seen as another success measure of the system.

Project progression

This project is naturally prone to a modular design and is likely too large to undertake in one step. It, therefore, made sense to do one step at time and produce a deliverable at each step. The first step of the project was the development of the visual model, and test it’s viability with self-other separation on a previously explored dataset. The results produced from looking a the real world dataset from Dr. Alexander Stoytchev’s dissertation had promising results showing that the visual model was viable for this purpose, at least for self-other separation in the real world.

The first step after the introduction of the game component was attempting to find an effective scheme to control the joystick. This was an imperfect process as a method was required that was both sufficiently random and effective in exploring most of the functional area of the joystick in a reasonable amount of random movements. Once this was done, it was hooked up to a sample game to make sure that technique was effective.

Once this was done, the two elements were combined with a large number of elements to fill in the blanks. The original implementation for this was done for a game called SDL ball that was implemented for linux in openGL. For this version of the project, the robot did not have to look off of a tv screen making it unnecessary to implement all the changes discussed above. Instead of a TV, the game was piped directly into memory shared between the game and the vision code.



A simplfiied version of an open-source breakout game called SDL ball was used for the demo during open lab night

This iteration of the game was seen as insufficient for a number of reasons. First of all, memory shared between the game and the self-other separation code could easily “cheat” and simply pipe “self” and “other” positions directly into shared memory and completely skip all the relevant parts of the process. To a somewhat lesser extent, The positions of objects could be sent into the self-other separation code bypassing the detection code entirely.

That state of the project was demonstrated at CPRE open lab night November 11, 2010. Shared memory was used to pipe images directly into the vision code but a television was displayed while the robot was playing in order to show the robot’s perception and progress.

The introduction of the television added a new set of milestones to be met. First of all, the original OpenGL-based game no longer met game requirements for effective tracking through a TV. An image tracked on a tv would have to have large and textured sprite so that they can be distinguished in a relatively noisy image. The new game underwent a number of tracking simulations before it was used in conjunction with a TV and the webcams. Quick testing revealed that an image with a white background would minimize reflections and make this whole process more feasible.

The final step in the game is ideally to play the game with the robot staring a television and controlling the game.

Sources

1.Gallup, G. G. J. Chimpanzees: self-recognition. Science, 167(3914):86–7, Jan. 2 1970.

2.Shaffer, D. W., Squire, K. D., Halverson, R., & Gee, J. P. (2005). Video games and the future of learning. Phi Delta Kappan, 87(2), 104–111.

3.Michel, P., Gold, K., & Scassellati, B. (2004). Motion-based robotic self-recognition. Presented at the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems.