Arm One *Little Man One* — *Delphi language* *Windows* — *Calculations*

THE CONTROL SYSTEMS IN ARM VERSION 1, DELPHI REVISION 1

This program simulates a human arm reaching out to touch a target the user can move in three dimensions. The arm has three degrees of freedom (two at the shoulder and one at the elbow). The position of the "fingertip" is ray-traced to form two retinal images in which both the target and fingertip positions appear. These images are used to derive x, y, and distance signals, which are controlled by a visual system that varies the reference signals entering the three kinesthetic higher-order control systems.

Comment on this Delphi version

The "Arm One" or "Little Man One" program was programmed by Bill Powers in 1989 as a DOS program using the C programming language. This program and its documentation is available for anyone interested. By 2004 this program was running a bit fast on Windows computers in spite of a clock speed adaptation, and would run in Full Screen only on a Windows XP computer, so Bill has undertaken to update the program using the Delphi (Windows) programming environment, reusing some of the original code and simplifying, rewriting some. Bill's writeup on the following pages, like most of Bill's writings, is not a modification of prior writing, but is written from scratch. Thus additional insight can be gained by studying the prior DOS version as well. The illustration featured in the prior documentation is included in this document, even though the writeup does not refer to it.

Commenting on the original program in a private email on May 2, 2004, Bill said: It was written in the early 90s, as I remember, in C [1989 it would appear]. I seem to recall that there was a model of a pointing arm published in Science around then, and I thought it was terrible and decided to make my own.

Note:

If you create a completely unusable set of parameters, just delete the "params" file and restart the program. A default set of parameters is then used and a new "params" file is created. Also, don't forget the reset button, which recovers from most awkward positions. If the display goes wild, readjust the parameters to low values, move the target away from the body, and hit reset. That should cure most problems without having to leave the program.

If you want to save a particular set of parameters, just rename the "params" file. A new one will be created. When you want to use the saved one, copy it to "params," overwriting the existing file.

INTRODUCTION

There are two levels of control in ArmV1DR1. The upper level controls the visual relationship between the fingertip and the movable target. The lower level controls individual joint angles and head angles.

LEVEL 1 CONTROL SYSTEMS

This level contains three independent control systems controlling joint angle in three degrees of freedom: shoulder azimuth or X direction (subscript a), shoulder pitch or Y direction (subscript b), and external elbow angle or Z direction (subscript c).

The "shoulder yaw" control system determines the azimuth of the arm. The reference signal *r1a* is limited to keep the arm within bounds on startup, and the perceptual signal *p1a* represents the actual azimuth of the arm, *AZ*. The error signal is passed through a leaky integrator with gain *g1a,* scaled up by 100 to increase the dynamic range of the integer calculations, then scaled down again when the output signal *o1a* is computed. Here is the source code for this control system:

```
procedure azimuth1;
begin
 if r1a > 768 then r1a := 768
 else
 if r1a < -768 then r1a := -768;
 p1a := az;
 e1a := r1a - p1a;
 i1a:= i1a + (100*g1b*e1a - i1a) div s1a;
 o1a := i1a div 100;
end;
```

In this early version of the model, there were no physical dynamics, so the actual azimuth of the arm was simply made equal to the output of the azimuth control system. The arm was working in the "imagination mode" in this version. In the later version 2, this imagination shortcut was opened up and a two-level kinesthetic control system with an arm having mass was inserted, making the overall model more realistic. A Delphi version of that model will be produced later.

The shoulder pitch control system is organized identically:

```
procedure vertical1;
begin
 if r1b < -1024 then r1b := -1024
 else
 if r1b > 1024 then r1b := 1024;
 p1b := t1 + 400;
 e1b := r1b  - p1b;
 i1b := i1b + (100*g1b*e1b - i1a) div s1b;
 o1b := i1b div 100;
end;
```

At the first level, the remaining controller controls the external angle at the elbow. Limits applied to the reference signal make sure the arm will not be hyperextended at the elbow, nor flexed with the forearm more than 45 degrees in the direction of the shoulder.

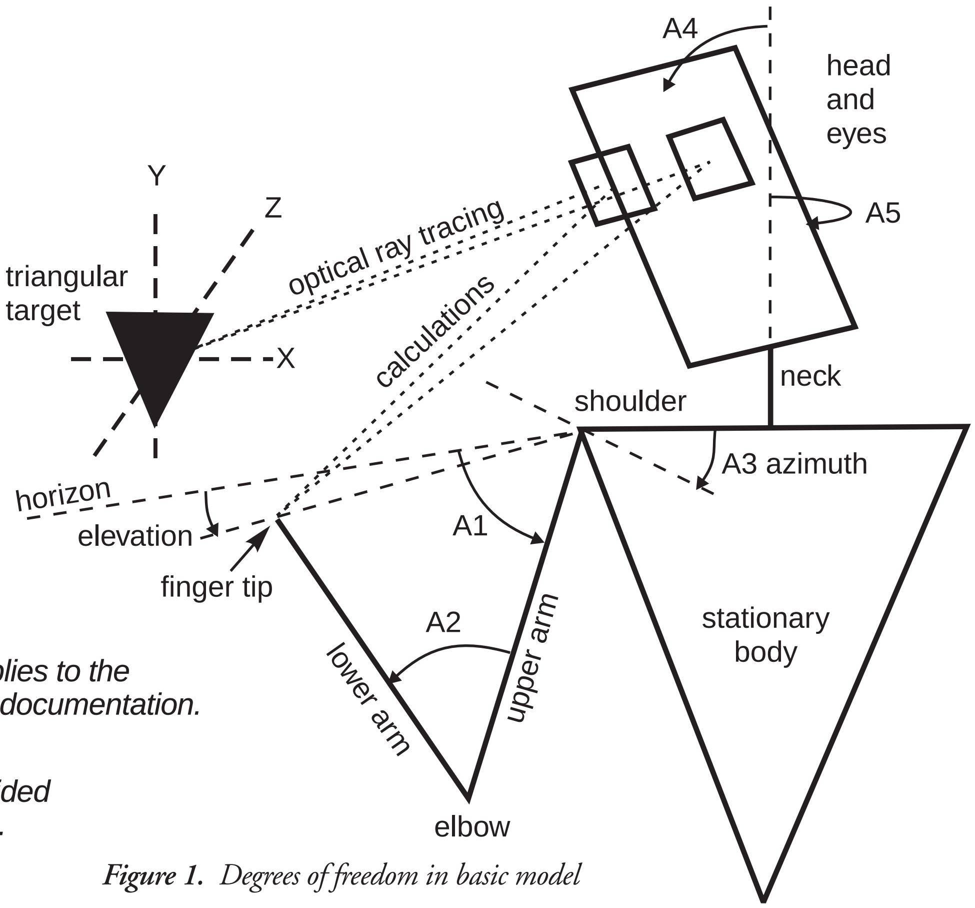
```
procedure distance1;
begin
 if r1c > 768 then r1c := 768
 else
 if r1c < -1024 then r1c := -1024;
 p1c := t2 - 768;
  {kinesthetic perception of shoulder-finger
  distance}
 e1c :=  -r1c - p1c;
 i1c := i1c + (100*g1c*e1c - i1c) div s1c;
 o1c := i1c div 100;
 end;
```

There are two additional "gaze" control systems at the first level which keep the image of the target centered in the right eye by turning the head in the X and Y directions. In effect, these two systems foveate the image of the target in the right eye. The later Version 2 foveated the target on the two foveas independently, so the eyes converged and diverged when the distance to the target changed. In this present version, the eyes gaze in parallel and only the head moves.

These gaze control systems keep the eyes oriented toward the target, which lessens the effects on angle measurements for depth perception when the objects are off the direction of gaze.

```
procedure gaze1x;
begin
 r1d := 0;
 p1d := TRx;
  {perceived x target deviation from gaze
  angle}
 e1d := r1d - p1d;
 i1d := i1d - e1d;
  {Note: pure integral control system}
 o1d := dmd(i1d,g1d,30);
end;

procedure gaze1y;
begin
 r1e := 0;
 p1e := TRy_;
  {perceived y target deviation from gaze
  angle}
 e1e := r1e - p1e;
 i1e := i1e - e1e;
 o1e := dmd(i1e,g1e,30);
end;
```



This figure applies to the DOS program documentation.

No figure provided for this writeup.

Figure 1. *Degrees of freedom in basic model*

Angles:

A1: Vertical angle between upper arm and horizon. (Positive above horizon. Negative as shown).
A2: Inside elbow joint angle in vertical plane. Always positive. (Shoulder to fingertip angle in vertical plane. Not shown. Computed = A1 + 1\2(180-A2). (180-A2) = outside elbow angle).*
A3: Horizontal angle between arm and body. (Zero when arm is straight ahead. Positive to the right).
A4: Vertical rotation angle between head and body. (Zero when head upright, negative when head tilts down).
A5: Horizontal angle between head and body. (Zero when head straight ahead, positive when turned to the

The links to the outside world are very simple in this version. The outputs of the five first-order control systems are routed back to the inputs as if lower control systems had instantly made the perceptions of the controlled variables match reference signals set by the output signals. This is where lower systems with physical dynamics are inserted in Version 2.

```
procedure link1to0;
begin
 t1 := o1b;       {vertical}
 az := o1a;       {azimuth}
 t2 := o1c;       {distance}
 gazex := o1d;    {gaze x}
 gazey := o1e;    {gaze y}
end;
```

LEVEL 2 CONTROL SYSTEMS

The second level of control consists of three independent control systems, two acting in azimuth and elevation, and one in depth.

The azimuth control system uses measures of Target and Finger direction as seen from the right eye. The Finger Right-eye x angle *FRx* is subtracted from the Target Right-eye x angle *TRx* to yield a perceptual signal *p1a* representing the angular separation of finger from target. This perceptual signal is subtracted from the reference signal, normally zero, *r1a*, to yield the error signal. The error signal is integrated to produce the output signal *o2a*, which becomes the reference signal for the azimuth control system at level 1 discussed above.

Note the "if purpose" statement. If a boolean variable *purpose* is set true, a sine- and cosine- wave generator is applied to the reference inputs for position in azimuth (x) and elevation (y). This results in the fingertip spontaneously tracing a circle around the target as seen by the right eye, regardless of target position.

```
procedure azimuth2;
begin
 if purpose then
  r2a := round(120*cos(elapsedtime))
 else r2a := 0;
 p2a := TRx - FRx;
 e2a := r2a - p2a;
 i2a := i2a + g2a*e2a;
 o2a := -e2a div 100;
end;
```

The elevation control system works the same way. Note that the variable called TRy, target right-eye y position, is terminated by an underline character. This is because "TRY" is a Delphi key word.

```
procedure vertical2;
begin
 if purpose then
  r2b := round(120*sin(elapsedtime))
 else r2b := 0;
 p2b := FRy - TRy_;
 e2b := r2b - p2b;
 i2b := i2b + g2b*e2b;
 o2b := g2b div 100;
end;
```

The last of the level-2 control systems controls the radial distance of the fingertip from the target, using approximate binocular depth perception. If d is the distance to the object, then the angle between the lines of sight from the two eyes to the object is approximately the interocular separation REx - LEx, divided by the distance d to the object.

```
disparity_angle ~ (REx - LEx)/d.

Thus the distance to the object is approxi-
  mately

d = (REx - LEx)/disparity_angle.
```

This control system perceives and controls the *radial distance* between target and fingertip, which is the difference between two depth perceptions. If TLRd is the Target Left-Right disparity angle, and FLRd is the Finger Left-Right disparity angle, the perceptual signal representing the radial distance between fingertip and target is

```
p1c = (REx - LEX) * (1/TLRd - 1/FLRd)
```

The requirements of integer arithmetic introduce some scaling factors, and make use of the "dmd" or "double-multiply-divide" routine, which multiplies the first two arguments together and divides the product by the third argument without losing precision.

```
procedure distance2;
begin
 r2c := 0;
 p2c := dmd(REx - LEx,4096,TLRd) - dmd(REx -
   LEx,4096,FLRd);
 e2c := r2c - p2c;
 i2c := i2c + e2c;
 o2c := -dmd(i2c,g2c,100);
end;
```

Finally, there is a simple linkage routine connecting the second-level output signals to the reference signals of the first-level systems. The azimuth system connection is straightforward, but there is a complication in the way the output of the depth-control system connects to the lower systems. The output of the second-order elevation control system, o2b, connects to the reference input of the first-order elevation control system, r1b. However, the output of the second-order depth control system not only connects to the first-order elbow angle controller, but also adds, with a weight of -0.5, to the first-order elevation controller's reference signal. The result is that an output from the second-order binocular depth control system extends the forearm at the elbow joint and simultaneously raises the shoulder angle. Since the two arm segments are equal, this results in a purely radial motion of the tip of the arm, the "fingertip."

```
procedure link2to1;
begin
 r1a := o2a;
   {finger azimuth}
 r1b := o2b + o2c div 2;
   {finger pitch }
 r1c := o2c;
   {finger distance (elbow angle)}
end;
```

OVERVIEW OF SIMULATION

There is a procedure called “environment” which computes the location of the fingertip from the current joint angles and arm segments, and then computes the various angles from the eyes to target and fingertip—the “ray-tracing” part of the program. This provides the necessary values of sight-angles for the control systems above. The “environment” procedure and the eight control system procedures are executed one after the other on each iteration of the simulation, advancing the variables by a small amount each time. Also, the variables are plotted and transcribed on the screen each time. By far the greatest portion of the program is devoted to displaying the variables and responding to user actions. Delphi allows the iterations to be done at a controlled rate, nominally 45 frames per second, independent of processor speed.