Trajectory Planning for Humanoid Robots

Matthew Burke, msburke20@students.polytechnic.org
Polytechnic School, Class of 2020
USC Viterbi Department of Robotics Engineering, SHINE 2018

Introduction

Dr. Schaal's Computational Learning and Motor Control Lab focuses on creating bipedal robots capable of navigation and movement using an approach that is inspired by how humans interact with the environment. Under the mentorship of Harry Su, I worked on proving the equation for the optimal trajectory from a previous paper and implementing it in a C program, which controls a NAO robot. During my time at CLMC, I learned how to plan a trajectory and interface with the NAO robot.

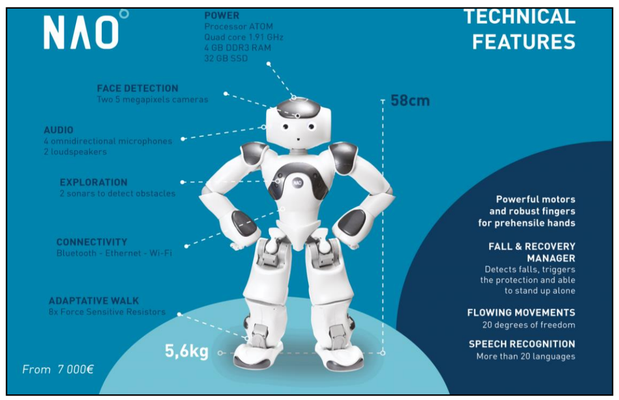


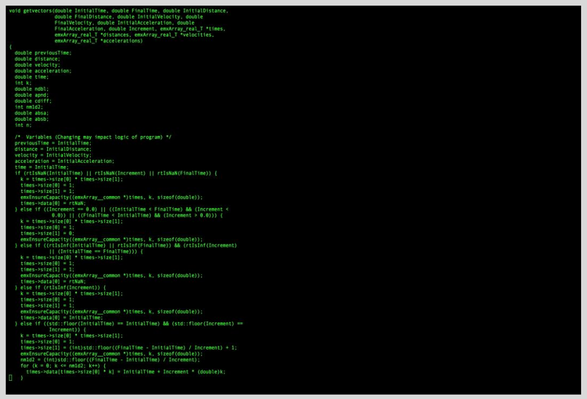
Figure 1. NAO Robot is a 58cm tall humanoid robot with 25 degree-of-freedom. It is one of the most advanced small scale humanoids with sensors for control and human robot interaction.

Objective & Impact of Professor's Research

Dr. Schaal's research is aimed at creating robots that can navigate unknown environments autonomously or with minimal human interaction. In order to achieve this, Dr. Schaal uses human guided learning techniques and looks towards how living organisms navigate. The future applications of this research are numerous, ranging from autonomous vehicles, transportation outside of roads, and military logistics.

Skills Learned

- Proved equation from Flash and Hogan paper and implemented it in MATLAB
- Learned advantages and limitations of current trajectory planning and reward functions
- Proved theorem about the coordination of arm movements in paper by using Linear Algebra, with Likelihood Function and Gaussian Elimination of matrices
- Wrote prototype in MATLAB and final program in C++



```
void getvectors(double InitialTime, double FinalTime, double InitialDistance,
                double FinalDistance, double InitialVelocity, double
                FinalVelocity, double InitialAcceleration, double
                FinalAcceleration, double Increment, emxArray_real_T *times,
                emxArray_real_T *distances, emxArray_real_T *velocities,
                emxArray_real_T *accelerations)
{
  double previousTime;
  double distance;
  double velocity;
  double acceleration;
  double time;
  int k;
  double ndbl;
  double apnd;
  double cdiff;
  double absa;
  double absb;
  double absb0;
  int n;

  /* Variables (Changing may impact logic of program) */
  previousTime = InitialTime;
  distance = InitialDistance;
  velocity = InitialVelocity;
  acceleration = InitialAcceleration;
  time = InitialTime;
  if rtIsNaN(InitialTime) || rtIsNaN(Increment) || rtIsNaN(FinalTime)) {
    k = times->size[0] * times->size[1];
    times->size[0] = 1;
    times->size[1] = 1;
    emxEnsureCapacity((emxArray__common *)times, k, sizeof(double));
    times->data[0] = rtNaN;
  } else if ((Increment == 0.0) || ((InitialTime < FinalTime) && (Increment <
              0.0)) || ((FinalTime > InitialTime) && (Increment > 0.0))) {
    k = times->size[0] * times->size[1];
    times->size[0] = 1;
    times->size[1] = 0;
    emxEnsureCapacity((emxArray__common *)times, k, sizeof(double));
  } else if ((rtIsInf(InitialTime) || rtIsInf(FinalTime)) && (rtIsInf(Increment)
              || (InitialTime == FinalTime))) {
    k = times->size[0] * times->size[1];
    times->size[0] = 1;
    times->size[1] = 1;
    emxEnsureCapacity((emxArray__common *)times, k, sizeof(double));
    times->data[0] = rtNaN;
  } else if (floor(Increment) == Increment) {
    k = times->size[0] * times->size[1];
    times->size[0] = 1;
    times->size[1] = (int)ndbl + 1;
    emxEnsureCapacity((emxArray__common *)times, k, sizeof(double));
  } else if ((InitialTime > 0.0) == (FinalTime < InitialTime) && (std::floor(Increment) ==
              Increment)) {
    k = times->size[0] * times->size[1];
    times->size[0] = 1;
    times->size[1] = (int)std::floor(FinalTime - InitialTime) + 1;
    emxEnsureCapacity((emxArray__common *)times, k, sizeof(double));
    for (k = 0; k <= ndbl; k++) {
      times->data[times->size[0] * k] = InitialTime + Increment * (double)k;
    }
  }
```

Figure 2. Selection of C++ Code

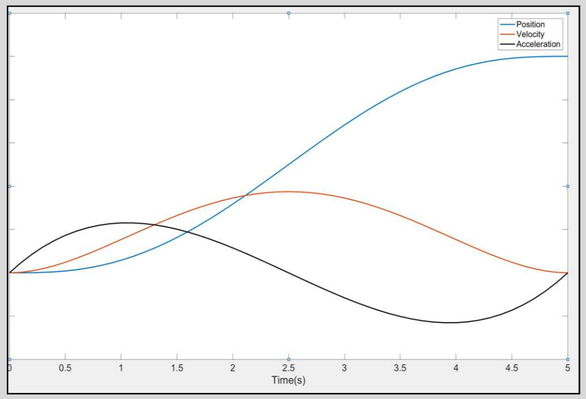


Figure 3. MATLAB graph of planned position, velocity, and acceleration vs time

Works Cited

Flash, T., & Hogan, N. (1985). The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neuroscience*, *5*(7), 1688-1703.

How This Relates to Your STEM Coursework

My work with the NAO robot gave me a better understanding of trajectory planning and some of the limitations of current approaches, such as the difficulty in correcting random errors, resulting in error compounding. Having a chance to practice working with MATLAB and C in a real lab situation will help me in AP Computer Science. Solving systems of equations using matrices will be useful in AP Calculus and future math courses.

Outside of my coursework at Poly, I spend 20 to 30 hours a week in independent study learning about current vulnerability exploitation methods. I have deep concerns about how the research and testing of security is not keeping up with the pace of technological innovation. I believe it is imperative that future computer scientists and engineers should place priority on securing new devices making their way into critical infrastructure. Therefore, I am pursuing study in information security and am now an Offensive Security Certified Professional. I am now seeking a certificate as an Offensive Security Certified Expert, which focuses on exploit development. In my spare time, I participate in "bug bounties" looking for weaknesses in software and I hope to publish a vulnerability report.

My interest in ethics was sparked by a college philosophy course I took at Davidson THINK at UNR and through participating in high school debate. I am concerned about how decisions are made by automated systems rather than by individuals or groups. Automation makes it almost impossible to assign responsibility when something goes wrong. I read philosophy and ethics in order to better understand the challenges that I will face in the future as an engineer or scientist.

CLMC Lab

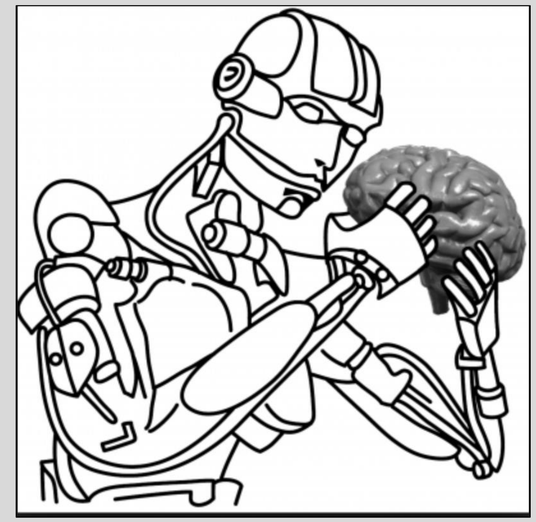


Figure 4. CLMC Lab Homepage Image - represents field of humanoid robotics

Next Steps

- Look for research opportunities in computer science
- Continue pursuing OSCE certificate from Offensive Security
- Apply knowledge obtained from this research experience to future projects
- Reach out to other professors in this field
- Continue to read academic literature in these fields
- Bring back knowledge learned to my school's robotics team and maker space club
- Continue to mentor students in robotics, programming, and math

Acknowledgements

- Dr. Stephan Schaal
- Harry Zhe Su – PhD candidate
- Dr. Katie Mills
- The SHINE Program