PHYSICOMIMETICS FOR MOBILE ROBOT OBSTACLE AVOIDANCE

Suranga Hettiarachchi[1] and William M. Spears[2]
[1]*Indiana University Southeast, New Albany, Indiana, USA*
[2]*Swarmotics LLC, Laramie, WY, USA*
[1]*suhettia@ius.edu,* [2]*wspears@swarmotics.com*

Keywords: Physicomimetics, Obstacle Avoidance

Abstract: The ability of robots to promptly avoid obstacles and reach a goal while accurately maintaining a formation is extremely important for many swarm robotic tasks such as terrain analysis, search and rescue, and chemical plume source tracing. In prior work, using simulation, we established how Physicomimetics can be used to maintain swarms of mobile robots in formations that move towards a goal through an obstacle field. In this paper, we demonstrate the obstacle avoidance capability of Physicomimetics control algorithm using two different mobile robot platforms with limited hardware, Maxelbot and X80Pro.

1 INTRODUCTION

The main contributions of this paper are: (1) deployment of Physicomimetics framework as the control algorithm of two different mobile robot platforms, (2) use of less complex implementation of Physicomimetics control algorithm for obstacle avoidance.

Our focus is to design, develop, and implement a swarm of robots that is capable of autonomous navigation towards a goal in an obstacle-laden environment while maintaining a formation. If the robots have limited sensor information and/or the environment changes while the robots are in the field, promptly avoiding obstacles and reaching a goal while accurately maintaining a formation becomes much more difficult.

We accomplish formation control and obstacle avoidance using techniques inspired by physical systems. We are motivated by physics because aggregate behaviors seen in classical physics are potentially reproducible with collections of mobile robots. We use our understanding of classical physics to derive the collective behaviors for robots. We do not restrict ourselves to copying physics precisely, so modifications are possible. The Physicomimetics framework is distributed, adaptive, cost effective, and rapidly deployable (Spears et al., 2004; Hettiarachchi and Spears, 2006).

The Physicomimetics algorithm presented here is less complex compared to majority of other robot control algorithms. Our algorithm is ideal for mobile robot platforms with limited hardware capability, and also directly protable from simulation to hardware with only few modifications. In addition, Physicomimetics approach consists of a theoretical framework that can be used to formally explain the robot behavior. We believe these are some of the major benefits of our algorithm.

In this paper, we focus only on the obstacle avoidance capability of Physicomimetics control algorithm using Maxelbot and X80Pro mobile robot platforms. Our work with X80Pro are at early stages, and we present the preliminary results. The rest of this paper includes an introduction to Physicomimetics algorithm with Hook's law as the force law, discussion on Maxelbot and X80Pro robot platforms, experimental methodology, results and analysis of the robot obstacle avoidance, related work, and conclusion with future work.

2 PHYSICOMIMETICS

In physics based approaches, virtual physics forces drive a agents and robots to a desired configuration or state. The desired configuration is one that minimizes overall system potential energy, and the system acts as a molecular dynamics ($\vec{F} = m\vec{a}$) simulation (Spears et al., 2004).

"Physicomimetics" or artificial physics (AP) is motivated by classical physics. This approach pro-

vides excellent techniques for distributed control of large collections of mobile physical agents as well as theoretical foundations for analyzing swarm behaviors. The Physicomimetics framework provides an effective basis for self-organization, fault-tolerance and self-repair of robot control (Spears and Spears, 2011).

In Physicomimetics, each robot has position $\vec{p}$ and velocity $\vec{v}$. We use a discrete-time approximation of the continuous behavior of the robots, with time step Δt. At each time step, the position of each robot undergoes a perturbation $\Delta\vec{p}$. The perturbation depends on the current velocity, i.e., $\Delta\vec{p} = \vec{v}\Delta t$. The velocity of each robot at each time step also changes by $\Delta\vec{v}$. The change in velocity is controlled by the force on the robot, i.e., $\Delta\vec{v} = \vec{F}\Delta t/m$, where m is the mass of that robot and $\vec{F}$ is the force on that robot. F and v denote the magnitude of vectors $\vec{F}$ and $\vec{v}$. A frictional force is included, for self-stabilization.

Our control algorithm *AP-lite (i.e. artificial physics lite)* uses Physicomimetics with Hooke's law (ie. $\vec{F} = -K\Delta\vec{p}$) as our force law. The AP-lite algorithm is provided in the APPENDIX. The Physicomimetics framework allows for self-organizing swarms, while AP-lite is only capable of avoiding obstacles and maintaining a formation. AP-lite is specifically designed as a leader-follower algorithm; only the leader broadcasts a signal, rather than all robots as in the Physicomimetics approach. The advantage of AP-lite over Physicomimetics is that AP-lite provides us with faster robot speed, and further, we can use theory to set the parameter settings. AP-lite computes the amount of power required for the left and right motors based on the forces acting upon the robot. Here, we present our AP-lite algorithm for obstacle avoidance with both repulsive and attractive forces acting on the robot.

The attractive goal force is a global component that is always active; this drives the robot forward with an equal amount of power to both motors. When the robot reaches an obstacle, AP-lite computes the repulsive forces acting on the robot, and change the power supply to the motors. If the robot senses an obstacle from the right (i.e. right-most sensor, $S0$, (see Figure 1) reads a high value), AP-lite reacts to this repulsion by supplying less power to the left motor. If the robot senses an obstacle from the left (i.e. right-most sensor, $S3$, (see Figure 1) reads a high value), AP-lite reacts to this repulsion by supplying less power to the right motor. If both sensors in the middle, $S1$ and $S2$, are reading high values, the repulsive forces from both sensors counteract the goal force, causing the robot to come to a halt. Our robots are nonholonomic and they always move in the forward direction.

The virtual angles of the positioning of the four IR sensors are shown in Figure 1. These sensor angles allow us to effectively model the robot's stopping behavior when all sensors detect an obstacle in front of the robot. We may or may not use all available sensors in our experiments presented here.

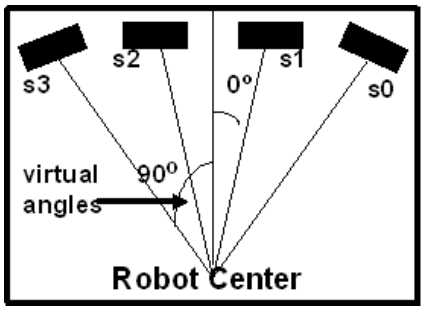

Figure 1: Positioning of sensors on the front of a Maxelbot.

We have deployed the Ap-lite obstacle avoidance algorithm using two different mobile robot platforms; the Maxelbot plat form designed by University of Wyoming and X80Pro platform designed by Dr. Robot Inc.

3 ROBOT PLATFORMS

The Maxelbot, as shown in Figure 2, by University of Wyoming is a modular low cost platform . The wheel base is an MMP5, made by The Machine Lab [1]. A primary MiniDRAGON is used for control. It communicates via an I^2C bus to all other peripherals, allowing users to plug in new peripherals as needed. The primary MiniDRAGON is the board that drives the motors. It also monitors proximity sensors and shaft encoders. A trilateration module is provided for robot localization (Maxim et al., 2008). The obstacle avoidance module consists of four GP2D12 IR sensors and the AtoD converter. The AtoD module converts the output voltage to a digital signal that consists of a raw sensor reading between a value of 0 and approximately 550 (the input voltage can be inconsistent), with 0 being the nonexistence of an object in front of the sensor and 550 being an object closest to the sensor.

The X80Pro robot hardware framework developed by Dr.Robot is a off the shelf product for researchers that offers full WiFi (802.11g) wireless with dual serial communication channels, multimedia, sensing and motion capabilities (Robot, 2012). The hardware framework contains two 12 volts motors, seven inch

[1] See http://www.themachinelab.com/MMP-5.html

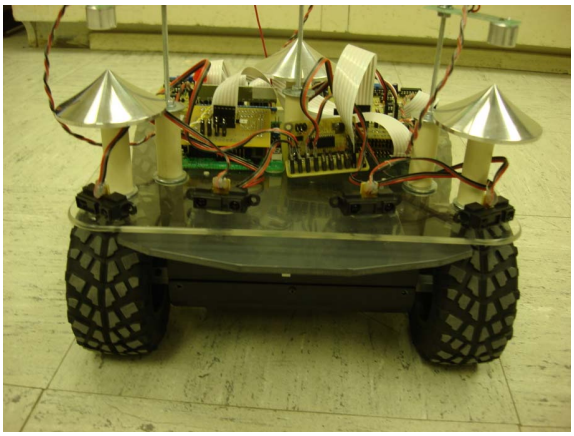

Figure 2: Front view of Maxelbot with IR sensors in the front.

driving wheels, DC motor driver module with position and current feedback, six ultrasonic range sensors, seven Sharp infrared distance measuring sensors, one pyroelectric human motion sensor, color image camera, and audio modules (speaker and microphone), sufficient hardware components to serve in variety of applications. The Figure 3 shows the front and back views of the X80Pro robot with all the components. In X80Pro robot, the IR sensor readings are measured between value of 0 and 80, with 80 being nonexistence of an object in front of the sensor and 0 being an object closest to the sensor.

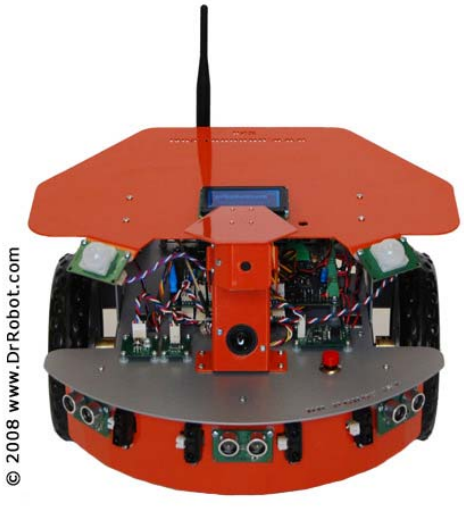

Figure 3: Front view of X80 pro with IR sensors in the front.

Figure 4: The X80Pro view of the environment. Light source at the far end to the left.

4 EXPERIMENTAL METHODOLOGY

All experiments with the Maxelbots are done outside with an unstructured obstacle environment. The outdoor environment consists mostly of grass, of average height 8 cm (3″), interspersed with concrete sidewalks, trees, rocks, leaves, and other debris. The grass hits the bottom of the Maxelbot. Although generally flat, the ground slope can change rapidly (within 61 cm or 2′), by up to 20°, at boundaries. We have placed large obstacles at random locations in the path of the Maxelbot mainly to see the effect in our data. We utilize all four IR sensors in the experiments with Maxelbot robot (See Figure 1).

All experiments with X80Pro are done in an indoor structured environment. The Figure 4 shows the robot view of this environment. Our work with X80Pro are at early stages and we are in the process of testing the robots in unstructured outdoor environments. The robot environment for X80Pro is modeled with two parallel walls and four obstacles in between with sufficient space for the robot to navigate. The goal, a light source, is kept at the far end of the two walls and all experiments are conducted in dark room. We do not make use of filters with X80Pro to filter out the noise in the environment. The robot is initially positioned at the bottom right corner of the Figure 4, behind the first obstacle to the right. Currently, we only utilize only the two outer most IR sensors in the experiments with X80Pro robot (See Figure 1).

All the results of our experiments are averaged over ten independent runs. All the experiments are conducted with a single robot.

5 RESULTS AND ANALYSIS

The following two sections provide results of the two robots and a discussion. The section 5.0.1 shows the results for Maxelbot robot, and the section 5.0.2 shows the results for X80 Pro robot. The Maxelbot results are analyzed using XY-scatter plots, and our preliminary results of the X80Pro robots are analyzed using line graphs.

5.0.1 Maxelbot Obstacle Avoidance Results

The Figures 5 through 7 show the corresponding relationship between the distance to obstacles and the power to Maxelbot motors using scatter plots. We use scatter plots because they provide a broader picture of the data. For all the graphs, the X-axis represents the distance to obstacles in inches and the Y-axis represents the power to the motor(s). The distance to the obstacles is measured using the function given below.

```
float ConvertSensorReadingToInches(rawIRVal)
     float temp;
     temp = (6787.0/(rawIRVal - 3.0)) - 4.0;
     return (temp/2.54);
end function
```

All the experiments are conducted in an outdoor setting. If the robot is less than 30 inches away from an obstacle, the robot feels the repulsive forces from that obstacle. Beyond 30 inches, obstacles have no effect on the robot. The closer the robot gets to an obstacle, the higher the repulsive force it feels. When the robot's repulsive forces overcome the attractive goal force, the robot comes to a stop.

The Figure 5 shows the distance to an obstacle on the right of the robot and the power to the left motor. There are five different scenarios visible in this scatter plot. First, the robot's left motor moves at a maximum constant speed of 70 when there are no obstacles repulsing the robot from the right. Second, the robot's left motor power decreases when the robot detects an obstacle from its right at a distance less then 30 inches. Third, the robot is even closer to the obstacle causing the middle-left sensor to detect the obstacle and this further decreases the power to the left motor. Fourth, the robot's left motor moves at the maximum speed even though the robot is detecting an obstacle on its right. This is because the robot reacts to a closer obstacle on the left before it reacts to the obstacle on the right. Fifth, the robot's left motor is stopped when there are no obstacles detected on the right. This is because the robot's two middle sensors detect an obstacle in front of the robot.

The Figure 6 shows the distance to an obstacle on the left of the robot and the power to the right motor. The scenario the robot faces is symmetric to the scenario seen in the previous (see Figure 5) graph. Interestingly, the robot detects less obstacles on its left compared to its right. This is caused by the difference in obstacle density along the robot's path.

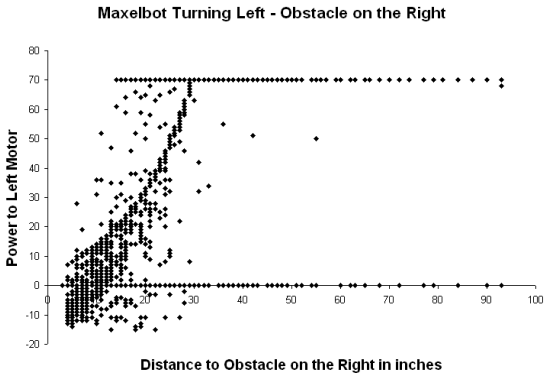

Figure 5: Correlation between the right-most sensor of Maxelbot, $S0$, and the power to the left motor.

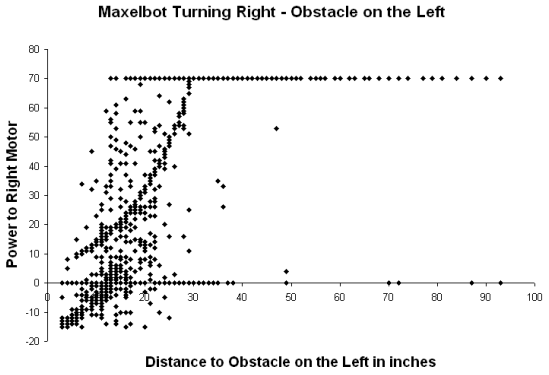

Figure 6: Correlation between the left-most sensor of Maxelbot, $S3$, and the power to the right motor.

The Figure 7 shows the average (the distances measured by the two middle sensors are averaged) distance to an obstacle in front of the robot and the average power to the left and right motors. The scatter plot clearly shows a linear correlation of the motor power to the left and right motors with the distance to an obstacle in front of the robot. The points where the distance is greater than 30 while the power is reduced correspond to situations where there are obstacles to the left or right.

5.0.2 X80Pro Obstacle Avoidance Results

The Figure 8 shows the infrared sensor readings during navigation of the X80Pro robot. The y-axis represents the raw sensor reading and the x-axis represents the time. The 370 time steps in all graphs approxi-

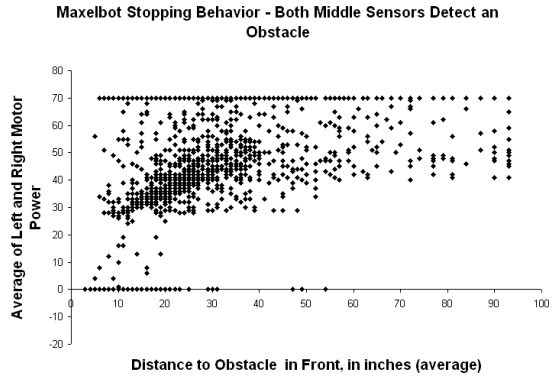

Figure 7: Correlation between the two middle sensors of Maxelbot, $S1$ and $S2$, and the power to the left and right motors.

mately correlates to 4 minutes of real time. The results clearly show the robot reaching the first obstacle to the right where the right most infrared sensor reading decreases at times between 125 and 175, and the robot reaching the last obstacle to the left before the goal at time 200 and 260. This behavior can clearly be seen in the power supply to the motor in the Figure 9 and the change in robot angle in the Figure 10.

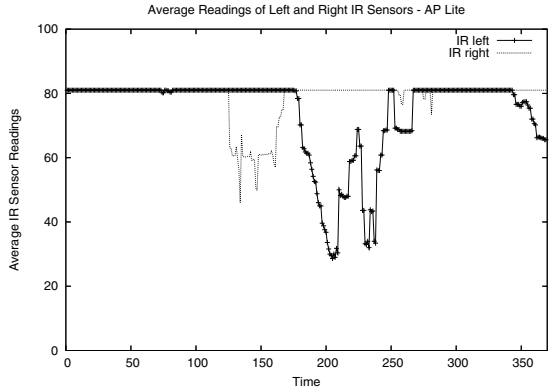

Figure 8: The raw IR sensor readings of the X80Pro robot during obstacle avoidance.

The Figure 9 shows the power to motors during navigation of the X80Pro robot. When the robot reaches the first obstacle to the right, repulsive forces are high from the right side of the robot. The resulting force vector causes the robot to reduce power to the left motor, but increase power to the right motor, allowing the robot to take a quick and sharp turn. Since this sharp turn causes the robot to keep an easy focus on the goal robot does not reach the last obstacle to the right, but detects the last obstacle to the left before the goal. To avoid the obstacle to the left of the robot at time 180, AP-lite control algorithm reduces the power to right motor while keeping the power to left motor the same.

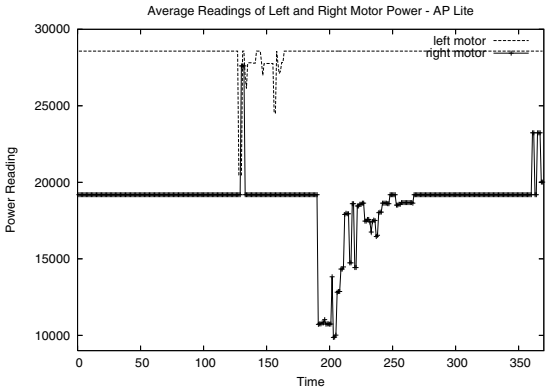

Figure 9: The power use by left and right motors of the X80Pro robot during obstacle avoidance.

We also measured the turning angle of the X80Pro robot, since AP-lite computes the turning angle based on the attractive and repulsive forces exerted on the robot by obstacles and the goal. The Figure 9 shows the turning angle during navigation of the robot in the y-axis over time in the x-axis. Once again, it is apparent that the robot's left turn occurs with the first obstacle to the right, and the robot's right turn occurs with the last obstacle to the left before the goal. We believe that the robot force vector computation favors the least resistant path from the starting point to the goal.

6 RELATED WORK

Both behavior-based and rule-based systems have proved quite successful in exhibiting a variety of behaviors in a heuristic manner. Fredslund and Matarić studied the problem of achieving global behavior in a group of distributed robots using only local sensing and minimal communication, in the context of formations (Fredslund and Matarić, 2002). The key concept of their algorithm is that each robot follows a designated "friend" robot at the appropriate angle and distance using a proximity sensor that can provide the angle and distance information of the friend. By panning the sensor appropriately, the algorithm simply keeps the friend centered in the sensor's view. They presented their results using four and eight robots in different formations. Balch and Arkin accomplished robot formations using the following two step process: "detect-formation-position" which is a perceptual process that determines the robot's position in the formation based on the current environment data, and

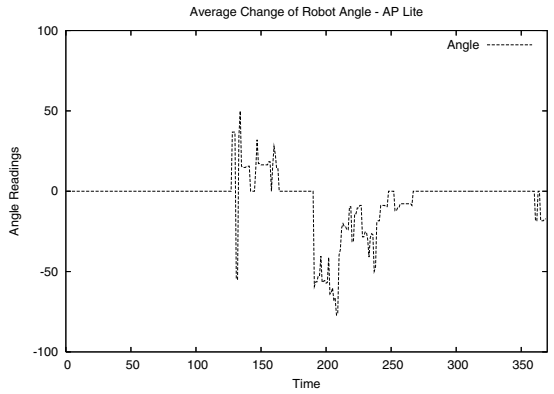

Figure 10: The turning angle of the X80Pro robot during obstacle avoidance.

"maintain-formation" which generates motor commands to direct the robot towards the correct location (Balch and Arkin,).

One of the earliest physics-based techniques is the *potential fields* approach (e.g., (Khatib, 1986)). Most of the PF literature deals with a small number of robots (typically just one that is similar to our experimental setup) that navigate through a field of obstacles to get to a target location. The environment, rather than the robots, exert forces. Obstacles exert repulsive forces while goals exert attractive forces (Kim and Khosla, 1991; Koren and Borenstein, 1991).

The *social potential fields* (SPF) framework is highly related to Physicomimetics framework (Reif and Wang, 1998). Reif and Wang rely on a force-law simulation that is similar to the Physicomimetics approach, allowing different forces between different robots. Their emphasis is on synthesizing desired formations by designing graphs that have a unique potential energy (PE) embedding.

7 CONCLUSION AND FUTURE WORK

In this paper, we demonstrate the obstacle avoidance capability of Physicomimetics control algorithm, AP-lite, using two different mobile robot platforms, maxelbot and X80Pro. We introduced our robot platforms, a novel control algorithm, AP-lite, for robot control. AP-lite is reactive to an attractive force from a virtual goal and repulsive to forces from obstacles. we have presented an analysis of our results using scatter plots. Our results demonstrate the accuracy and quality of both our hardware and software.

Future work of this research will focus on extending the implementation of our algorithms to multi-robot systems since the algorithms are already theoretically extended to handle swarm of robots. This will also allow us to make use of techniques presented in (Reif and Wang, 1998). Another improvement would be to implement filters to eliminate noise in the environment and test the robot behavior in more complex environments.

REFERENCES

Balch, T. and Arkin, R. Behavior-based formation control for multi-robot teams.

Fredslund, J. and Matarić, M. (2002). A general algorithm for robot formations using local sensing and minimal communication. In *EEE Transactions on Robotics and Automation*, volume 18, pages 837–846.

Hettiarachchi, S. and Spears, W. M. (2006). Daedalus for agents with obstructed perception. In *Proceedings of the 2006 IEEE Mountain Workshop on Adaptive and Learning Systems*, pages 195–200. IEEE Press.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In *International Journal of Robotics Research*, volume 5(1), pages 90–98.

Kim, J. and Khosla, P. (1991). Real-time obstacle avoidance using harmonic potential functions. In *IEEE International Conference on Robotics and Automation*, pages 790–796.

Koren, Y. and Borenstein, J. (1991). Potential eld methods and their inherent limitations for mobile robot navigation. In *IEEE International Conference on Robotics and Automation*, pages 1398–1404.

Maxim, P., Hettiarachchi, S., Spears, W., Spears, D., Hamann, J., Kunkel, T., and Speiser, C. (2008). Trilateration localization for multi-robot teams. In *International Conference on Informatics in Control, Automation and Robotics*. ISI Proceedings.

Reif, J. and Wang, H. (1998). Social potential elds: A distributed behavioral control for autonomous robots. In *Workshop on the Algorithmic Foundations of Robotics*.

Robot, D. (2012). Dr. robot inc - extend your imagination. `www.drrobot.com/products_item.asp?itemNumber=X80Pro`.

Spears, W. and Spears, D., editors (2011). *Physicomimetics-Physics Based Swarm Intelligence*. Springer, New York, 1st edition. This is a cross-referenced BOOK (collection) entry.

Spears, W. M., Spears, D. F., Hamann, J., and Heil, R. (2004). Distributed, physics-based control of swarms of vehicles. In *Autonomous Robots*, volume 17(2-3), pages 137–162. Springer.

APPENDIX

```
void ap lite()
    float v, vx, vy, r, F, kmid, kside, Fx, Fy, delta vx, delta vy
    float theta, theta new, delta, w, FR, STEP, mass, alpha
    int power left, power right
    int RANGE = 30 // reactive distance to an obstacle
    int MAX SPEED = 70 // maximum power supply to motors
    STEP = mass = 1.0 // step size and the mass of the robot
    kmid = 5.0 // Hooke's law constant for two middle sensors
    kside = 4.0 // Hooke's law constant two out side sensors
    FR = 0.5 // friction
    alpha = 1.0 // decides the amount to turn
    Fx = 100.0 // attractive goal force in x direction
    Fy = 0.0 // robot do not move side-way
    v = current velocity // initial value is at 70%
    vx = FR * v // velocity drops with friction
    vy = 0 // No side-way velocity
    for (all four sensors)
        if (sensor reading > 30) // filter out low readings
             r = RANGE - ConvertSensorReadingToInches(sensors reading)
        if (r < 0) r = 0 // no obstacles seen
        theta = virtual sensor angle in radians
        if ( sensor 1 or sensor 2)
             F = kmid * r // compute force from the two middle sensors
        else
             F = kside * r // compute force from the two out side sensors
        Fx = Fx - (F * cos(theta)) // Repulsive x component
        Fy = Fy - (F * sin(theta)) // repulsive y component
        endif
    endfor
    delta vx = STEP * Fx / mass // change in velocity in x direction
    delta vy = STEP * Fy / mass // change in velocity in y direction
    vx = vx + delta vx // velocity in x direction
    vy = vy + delta vy // velocity in y direction
    v = sqrt(vx * vx + vy * vy) // current velocity
    delta = atan2(delta vy, delta vx) // direction of change in velocity
    current velocity = v // reset current velocity
   w = turn function(delta) // angle of the robot turn
   theta new = atan2(vy, vx) // robot moves in first or second quadrant
    if ((-PI/2.0 <= theta_new) and (theta_new <= PI/2.5))
         power right = (int)(v + v * alpha * w) // power to right motor
         power left = (int)(v - v * alpha * w) // power to left motor
         // proportionally cap the motor power
         if (power right > MAX SPEED or dcl > MAX SPEED)
                if (power right >= power left)
                        power left = MAX SPEED * power left / power right
                        power right = MAX SPEED
                else
                        power right = MAX SPEED * power right / power left
                       power left = MAX SPEED
                endif
         endif
    endif
    if (dcr >= 0)
            move forward right motor with power right
    endif
    if (dcl >= 0)
            move forward left motor with power left
     endif
end ap-lite


float turn function (float angle)
    float angle radians = angle
    // robot moving to the second quadrant
    if ((PI/2.0 < angle) and (angle <= PI))
         angle radians = PI - angle
    endif // robot moving to the third quadrant
    if ((-PI < angle) and (angle < -PI/2.0))
         angle radians = -PI - angle // only used in backward move
    endif
    return angle radians
end turn function
```