**Implementation of a PID controller for Achieving Navigation Behaviours of a Mobile Robot**

*Adam Hubble (P17175774)*

***Abstract – Fundamental to the evasive behaviours of mobile robots, edge following is recognised for its application as an object avoidance and navigation strategy; to implement edge following behaviours, a proportional-integral-derivative (PID) controller can be adapted, for enabling a mobile robot to adjust its position to a given distance from an object, that’s edge, is calculated trackable. For the nature of this report, the configuration of a PID controller used to implement edge following behaviours, will be explored, for a mobile robot that exists in a structured environment.***

***Keywords – object avoidance, environment navigation, proportional-integral-derivative controller, mobile robot, edge following, structured environment***

I. Introduction

For the structured nature of the environment provided and the continuing deployment of the Pioneer P3-DX mobile robot [1], the configuration for the mobile robot controller presented, was expected to employ a proportional-integral-derivative (PID) feedback controller [2], for the purpose of enabling the robot to follow the edges of objects (walls) that are determined followable, at a set distance (set-point) over time. Both the determination of a wall being trackable and the expectation of a PID controller to incrementally adjust the robots distance from a given wall over time, aimed to be calculated by the distance readings returned by the array of “range-finding Sound Navigation and Ranging (SONAR) sensors”, that the robot features [3].

***Available in Appendix A***

Relating to the robot controller’s capability of appropriating actuation based upon the analysis of data returned by said sensors, it was expected for the robot to invocate edge following behaviours upon a wall being detected; where it would initially align parallel with the detected well and then actuate a forward mode of traversal that facilitates oscillatory adjustments to its orientation overtime, in attempt to settle the set-point configured, as its distance from the wall. Given the random wandering capabilities of the robot (configured in the controller submitted prior), it was again expected of the robot to follow the edges of walls detected on either of its sides, to cater for the fluidity (transition) between active states and prevent the robot from exhibiting undesired behaviours. An example of such behaviours would be turning unnecessarily, in result of the robot being configured to follow the edges of objects for one of its sides and a wall being detected on the side that’s unconfigured; this can potentialize collisions occurring, due to the robot’s wheels being unaligned with the centre of its exterior body [1].

***Available in Appendix B***

In further mention of walls and in focus of their arrangements within the environment provided, it was also anticipated for the PID controller’s configuration to be compatible with the robot maintaining a set distance from walls, that are perpendicularly connected; this was deemed necessary for the robot to remain to exhibit edge following behaviours, as opposed to invocating behaviours specific to the random wandering state. Whereas previously figured, walls aligning with interior and exterior-angles (relative to the detectable side of the walls) would encourage the robot to actuate an angular mode of traversal, as it is expected to navigate ‘around’ the walls, in the form of constantly maintaining a set distance from them. This concerns the robot’s requirement to remain following the faces of said walls, given that the angular traversal patterns exhibited by the robot, would potentialize itself from becoming out of the detectable-range with the wall being followed; in such scenario, random wandering behaviours would be expected to be invocated, provided that an alternative wall is not detected. For acknowledgment purposes, angular traversal was an expected behaviour of the edge following state, given that it was known to represent the “continuous updates” [4] issued by the PID controller, to the velocities of the robots motor components; said updates are made relative to the error (discrepancy in set-point and robot distance to wall) most recently calculated.

***Available in Appendix C***

Proceeding from the configuration of a loosely-functioning PID controller, the values of variables associated with the PID controller were expected to be fine-tuned manually by default, or optionally, compiled autonomously. The underlying purpose of modifying said values was assumed to “adjust the reactions of the PID controller to changes in the set-point and for minimising the variability of control error” [5]; this would translate to the robot maintaining the configured set-point faster, however the behaviours exhibited by the robot would not be realised, until it was observed and subjected to a range of walls within its environment. For the arrangements of walls previously mentioned, a dispute between the regularity of set-point adjustment and the desirability of the behaviours exhibited by the robot, was anticipated to be noticed throughout the tuning process of the PID controller. This was figured as the frequency of oscillatory motions exhibited by the robot during its initial and proceeding adjustments “around the desired set-point” [6], was known to increase exponentially with heightened values, given that the rate of response to a calculated error (proportional gain) would be enhanced.

1. *Software development*

As for the software application used to implement a robot controller featuring PID feedback control, CoppeliaSim [7] was reapproached for its directness and continuing support for development, of the robot controller established in the submission prior. Given the foundation of knowledge acquired from the ‘Mobile Robotics’ undergraduate module and recent use of the software application for the current module, remote application programming interface(s) (API’s) were once again deferred from, given their unfamiliarity in user interface (UI) and native programming language; thus, for the implementation of PID feedback control and the resulting development of the robot controller, CoppeliaSim (Lua) has been used independently.

1. *Pioneer P3-DX composition*

For identifying the sensing and actuation capabilities available to the PID controller’s implementation, it was deemed sensible to revise the mechanical composition of the Pioneer P3-DX robot, within the integrated development environment (IDE) being used.

Given the sixteen range-finding SONAR sensors and two-wheel two-motor differential drive [1] onboard the robot, it was figured that the robot could transmit, receive and transduce ultrasound waves directed at walls, into range-related data. From the reception of said data, the robot would then be able to calculate its position offset to a wall that it is parallel to and detecting, whilst attempting to adjust autonomously, to the set-point configured. Adjustment in the robot’s distance from a given parallel-detected wall, was acknowledged possible by alternating the velocity of the wheels motor that faces the detected wall adjacently; this was assumed for maintaining a forward mode of traversal during oscillation, where the opposing wheels motor would maintain a constant velocity, to become a temporary pivot-point of the robot. All of which is supported by the robots two-wheel two-motor differential drive, that allows either motor to be operated independently.

1. *Controller architecture*

Regarding the architecture for the robot controller submitted, the controller does not defer from the original design configured previously; this decision was satisfied by the controller’s robustness and lacking requirement to be revised, given the PID controllers existence prior to the assignment being issued. Thereby, both random wandering and edge following behaviours possess the means to subsume one another, from the upheld application of the subsumption architecture [8]; through which, the controller features a series of finite-state machines (FSM’s) comprised of state determining variables (Boolean) and conditioning statements, to control the activeness and interchangeability of the controllers major and subsidiary states.

However, extending from the configuration settled for the robot controller previously, an additional function has been implemented and dedicated to disconnecting the autonomous fine-tuning functionality of the PID controller, from the actuation functionality of the robot’s edge following behaviours. Given its contribution to testing procedures and not to the “increase in levels of competence” [9] for the controller, explicitly, separating the fine-tuning functionality for the PID controller was sensible for maintaining the organised state of the controller, wholesomely.

***Available in Appendix D***

II. PID Controller

Assuming the “transient and steady-state responses” [10] to error that PID controllers as “feedback control systems” [11] provide, their application for devising adjustment behaviours in wheeled mobile robots, dictates that the velocities of the robots motor components can be controlled progressively, to “nullify the error” between the set-point configured and distance from an object that the robot actually resides at. This is known by the application of the closed-loop feedback algorithm, in “90% of industrial controllers” [10] today; some of which are enrolled within “navigation systems” [11], where they can often be purposed for autonomously controlling the “motor drives” of “wheeled robots”. Therefore, in context of the robot controller submitted, the PID controller’s implementation was purposed as the fundamental edge following behaviour of the robot, where in use of feedback control, the robot is tasked to maintain a set distance from walls that it detects over time, whilst traversing with a forward trajectory in attempt of navigating from the “source location and reaching the destination location, with negligible error”.

In continued mention of the set-point, its value alongside the maximum distance from a wall that the robot can remain edge following at (maximum distance) and distance at which the robot cannot detect an object beyond (undetectable range), was configured with the purpose of maintaining a close distance to a wall, whilst enabling the robot to remain within the edge following state and without it exhibiting sizable oscillations in the adjustment to error. For which, oscillations were known to be the product of the robot detecting a wall prematurely in comparison to the set-point, or inversely, from the robot detecting a wall within a sizable margin of the set-point. This was figured in the testing process that was conducted for the prior implementation of robot controller, where the set-point of ‘0.25’ metres, maximum distance of ‘0.275’ metres and detectable range of objects at ‘0.35’ metres, were settled as the optimal values for the purpose described; whereas values tested greater and less than this configuration, promoted the exhibition of behaviours that were undesired due to their opposition to the purpose outlined. It was therefore sensible to reapply the values listed, for their compatibility and adherence to said behavioural expectations.

1. *Proportional controller*

Given by the name of the parameter, the proportional error of the PID controller is calculated to be “proportional to the actual error obtained”, which translates to the robot being able to return to the configured set-point from the wall it detects by adjusting the velocity of the motor, attached to the wheel that adjacently faces said wall. For its calculation, the error for the current timestep is required to be known, which can be “adjusted by multiplying with a constant, known as a proportionality constant” (proportional gain); the value of this gain variable can be predetermined or set dynamically during runtime, however, for the robot controller submitted, this value is predefined and remains static in correspondence to the testing procedures conducted. The values configured for the integral and derivative gain variables proceeding, also share this setup.

When calculating error, the robots front left and right-most facing sensor readings are sampled (sensors indexed ‘1’ and ‘8’), relative to the side of the robot that an object is detected on; this is due to the front side-most sensors forming adjacent relations with a wall when the robot is parallel to it. In which said sensors were most appropriate for determining the robot’s distance from a wall, simply, whilst enabling other sensors to facilitate collision detection ahead of the robots path; this was in knowing of each sensor being uniquely positioned and angularly offset, for the robots mechanical composite [12]. Using the range returned by the robot’s front side-most sensors whilst maintaining a detection with a wall, error can be calculated by subtracting the returned reading from the set-point value configured; upon the error being calculated, it is then multiplied by the proportional gain value and appended to the default velocity (‘0.2’ metres per second) of the corresponding motor. Typically, it is found that smaller proportional errors “result in slower performances” [11] of nullifying said error, whereas larger proportional errors “result in system instability” (oscillation); inevitably, the proportional gain value passed to the PID controller, required to solve a balance between the conflicting nature of the behaviours mentioned.

***Available in Appendix E***

1. *Integral controller*

Moreover, for the integral parameter of the PID controller, calculating integral error purposes to “speed-up the process of settling” to the set-point configured, which can be achieved by “accumulating the error calculated over time” and performing an averaging operation, that considers the number of errors accumulated (integral threshold). When compared to the proportional error, integral error benefits the controller by catering for the reduction in adjustments long-termly, which eliminates the “long-time steady-state error of the proportional parameter”. In regard to the robot, integral error translates into hastened rates of adjustment to the set distance from a detected wall, which assumes a higher density of oscillatory motions exhibited by the robot, for a given period of time; to note, the influx and scalability of said motions, is typically factored by “huge integral errors” being calculated.

When calculating integral error, the error determined for the proportional controller is stored into an array variable (table), for the current index of error calculated; this is addressed using an integer variable representing the count of errors stored, which is reinitialised upon reaching the integral threshold (array capacity) configured, to enable previously calculated errors to be overridden. Updating calculated error determines that the adjustment to motor velocity is made relevant to the latter series of errors calculated, thus allowing the robots adjustment to the set-point, to marginalise with error, overtime. For which, the integral threshold is decided as ‘10’ for considering the previous ‘10’ errors calculated, which is sensible for regulating newly calculated errors and discarding previous errors simultaneously; this configuration accommodates the robots task to follow ‘around’ the perpendicular sequencing of wall faces, where the offset in error between arched and linear motion paths of the robot, becomes significant more suddenly, when compared to the implementation of higher thresholds. Thereby, the configuration promotes an increased attraction between the robot and wall, where the adjustment to error increments exponentially, for maintaining a detection and navigable relation with a wall in the scenario portrayed. For all errors calculated and stored in the array, each is accumulated and stored as an integer variable, which is then divided by the integral threshold to determine the integral error, as an average. Upon the error being calculated, it alike the proportional error is then multiplied by its corresponding gain value, before being appended to the default velocity of the robots operating motor.

1. *Derivative controller*

Lastly, in focus of the derivative parameter of the PID controller, calculating derivative error provides accelerative adjustment to error, alike the integral parameter, however, it remains “proportional to the difference in the change of error at every instant”, as opposed to compiling an averaged change in error, for a given number of errors calculated. In which, derivative error only considers the difference in current and prior errors calculated, to determine. When relating to the implications on the robot behaviourally, derivative error also promotes an increased frequency of oscillatory motions for a given period of time, as the robots rate of adjustment to the set-point enhances with correspondence to the “sensitivity in change” of the current and previous calculated errors. Similar to the integral error, a derivative error considered high assumes that the robot will exhibit said motion, at increased frequencies and of larger magnitudes, when compared to errors less significant. Inevitably, sizable oscillations exhibited by the robot’s movements is undesired for the fluidity between behavioural state transition and for maintaining a detection and navigational state with a wall.

When calculating the derivative error, the error computed for the proportional error is recycled again and stored as a separate float variable that represents the current error. Meanwhile, the error calculated previously is also stored as a float variable and is set to the current error, upon the current iteration of the derivative error being calculated at the end of frame executing. Using the subtraction operator, the derivative error can then be determined by calculating the difference between the current and previous errors; upon the error being calculated, it aligned with both the proportional and integral errors, is then multiplied by its predefined gain value, before being affixed to the default velocity of the robots motor, that forms the adjacent relation with the wall detected.

Granted that the proportional, integral and derivative errors are calculated and appended to the default velocity of the affected motor, the robot is able to pivot to the set-point overtime by regulating said motors velocity proportionally to the cumulative output calculated, whilst the opposing motor maintains a constant velocity of ‘0.2’ metres per second, to warrant the pivot point of the robot and forward mode of traversal for its navigation around the environment.

III. Testing

Given the known “change in type of response of the system” [13] that “proportionality constants” provoke when varied, tuning the PID controller, “which is the variation of the PID proportionality constants, is of utmost importance” for enabling the robot to marginalise its adjustment to the set-point overtime, whilst preventing sizable oscillatory motions being exhibited for collision prevention purposes and lastly, to establish an adequate attraction between the robot and the walls that it detects, for enabling it to navigate around its environment, exclusive of interruption caused by the invocation of random wandering behaviours; this specifically applies to the perpendicular sequencing of wall faces, as discussed in prior sections.

When compiling the final configurations for said constants, both manual and autonomous modes of test procedures were conducted, to expand the potential of compiling the PID controller with constant values that optimise the robot to “achieve its multiple and conflicting objectives” [14]; this decision was assumed better than manual tuning alone, given its known “time-consuming process” [15] for testing with higher numerical fidelity, as opposed to the “minimal influence of the human factor when selecting the setting, running, or supervising the process” of auto-tuning.

1. *Manual fine-tuning*

For manually adjusting the values of the proportionality constants, a series of black-box test cases were conducted, given the methods support for behavioural and numerical-based observations of the robot, which allows the adjustments allotted to said variables to be acknowledged and measured visually: better or worse, relative to the desired output. Regarding the approach to manual tuning, the PID constants were tested in an exploratory nature [16], where the logic used to modify their values, dynamically adjusted with the behaviours exhibited by the robot; this was deemed sensible for approximating the optimal values of the constants, so that the robot’s adherence to its expectations could be bettered. Given the “propagation of ultrasonic waves” [17] discovered in applications of SONAR technology, for the numerous “active transmitters of acoustic waves of acoustic frequencies” that the Pioneer P3-DX equips [1], the values configured in both manual and automated based tuning procedures, are acknowledged as only being approximate to optimisation as opposed to being absolute, given the immediate imprecision of the robots hardware and the potential fidelity of the values, being infinite.

When tuning the values of the PID constants, the environment that the robot is originally exposed to was used to validate the relevancy and effectiveness of the behaviours exhibited by the robot, for the environment directly, in correspondence to the values passed and the behavioural expectations of the robot, that were charted prior to the testing process being conducted. This enabled the manual tuning process to be less time-consuming than what was forecast, given that an optimal configuration compiled for the robot in an environment separate to the original, would hinder the same configurations performance for the environment that the robot is originally tasked for; thus, it was assumed that the optimal values tuned for the PID constants in an external environment would be subject to re-configuration, due to the behavioural expectations of the robot not being fulfilled in the original. Therefore, the robot’s behaviours were evaluated and enhanced in the environment original to its tasking.

Aside from the observational assessment of the robot’s behaviours, for acknowledging the robot’s rate of adjustment to the set-point over time, precisely, root-mean-square error (RMSE) is calculated to represent the “marginalisation of variance” [18] (error) in a numerical and therefore, comparable format. For which, RMSE can be compiled by finding the square root, of the mean, of the sum of squared errors calculated whilst the edge following left or right subsidiary states are active. In a programmatical sense, RMSE is handled by accumulating every proportional error calculated, squared, and stored as a float variable; for every proportional error then calculated, a separate float variable increments to represent the count of errors calculated. Upon the counter incrementing, it is then applied as a denominator to the accumulated error variable, as part of the divisive operation which returns the resulting RMSE value, unrooted (the mean). Given the value returned, its square root is then retrieved and stored as the native RMSE value.

***Available in Appendix F***

For comparative purposes, a series of graph objects and console output was configured to represent the depreciation of error visually and numerically, over time. In which, a time graph object was configured for plotting the RMSE value of every frame passed whilst the robot follows and maintains a detection with the edge of a wall; in use of the error visualization, trends relative to error adjustment over time were more easily interpreted. This was also supplemented by the configuration of a separate time graph object, that exclusively represents the proportional error calculated over time. Meanwhile, in use of the IDE’s console output, the RMSE value was displayed in the interface for every frame passed, where it could be recorded in its standard format. To note, for the autonomous tuning algorithm, RMSE was also referred to, despite its initial purpose for guiding manual adjustment with the PID constant values.

***Available in Appendix G***

Proceeding, is the stepwise solution for the manual tuning procedure performed, where *Kp* represents the proportional constant (gain), *Ki* represents the integral constant and *Kd* represents the derivative constant for PID feedback control:

1. (*Kp*) Initialise to insignificant value
2. (*Kd*)Set to *Kp* multiplied by 100

***If*** *slow and small oscillation(s)* ***then***

(*Kd*) *Increase until fast and large oscillation(s) occur(s)*

***Else***

(*Kd*)Reduce by factor of 2

1. ***If*** *oscillation(s) occur(s)* ***then***

(*Kp*) Reduce by factor of 10, until oscillations halt

***Else***

(*Kp*) Increase by factor of 10, until oscillation(s) occur(s)

***Then***

(*Kp*) Reduce by factor of 2 of less until stability is achieved

1. (*Ki*) Initialize to high numerical fidelity value

***If*** *no oscillation(s) occur(s)* ***then***

(*Ki*) Increase by factor of 10, until oscillation(s) occur(s)

***Else***

(*Ki*) Reduce by factor of 10

1. Continue to fine-tune observationally

For the test cases supporting the manual tuning procedure, refer to ***Appendix H***.

1. *Automated fine-tuning*

Meanwhile, in focus of auto-tuning, an algorithm was developed within the IDE to appropriate the values for the PID constants autonomously, which adheres to an iterative procedure concerning randomisation and reasoning, for a prolonged period of execution (six-day period). As mentioned previously, autonomous tuning facilitates higher numerical fidelities whilst not requiring human intervention, which assumes that a larger yield of compatible values can be generated “in a fraction of time” [15], when compared to slowed manual refinement; whereby, it is also known for more “conventional tuning techniques to lack the intelligence and flexibility, which would increase the performance rate and also improvise the stability and error criterion” [14] for a given PID controller configuration. However, when relating to the establishment of such algorithm, auto-tuning requires more consideration when compared to manual processes, given its obligation to support the robot with “achieving its multiple and conflicting objectives”, that depends upon a learning process factoring the variation between the values of the proportionality constants.

Opposing the considerations devised for the manual tuning procedure, auto-tuning exposes the robot to an environment separate from the original, for the purpose of accelerating the rate of generating, appropriating, and resolving compatible configurations of the PID parameters. For which, the environment used to instrument the auto-tuning process is comprised of a series of walls, that each form numerous perpendicular arrangements to encapsulate the robot within a unspecified area, where it can maintain discovery and a resulting navigable state with the walls, for regulating feedback control faster, comparatively. Given that the robot is tasked to follow the walls around continually, at each perpendicular joint, the robot is expected to invocate edge following transition state behaviours, upon detecting a wall ahead of its trajectory, where it aligns itself parallel to the wall it detects before following the edge faced adjacent to it; for the transition revealed, each wall of the environment is used to signal the influx of adjustment to the value of the PID constant, that’s currently iterated for the reasoning process. When considering the algorithms application in the original environment, signalling adjustment to the PID controller would be disproportionate to the robot’s distance travelled whilst edge following, given that each border for the original environment is of variable length which potentializes difference in error marginalization and the frequency of oscillatory motions exhibited by the robot, for the same values configured. Thereby, unlike the manual tuning procedure, a separate environment with a symmetrical arrangement was required, for compiling configurations that offer comparability through repetition, which could be used to appropriate the adjustments allotted to the values in proceeding iterations of the algorithm, so that the robot’s adherence to its behavioural expectations could advance.

***Available in Appendix I***

In mention of the algorithm computed, auto-tuning initially functions to sample randomly generated numbers with randomly generated degrees of fidelity, from ranges specific to each proportionality constant. To address this, a range given between ‘0.001’ and ‘100’ is potentialized for every initial instance of constant value to be tuned, which is calculated by generating a number between the ‘1’ and ‘100000’ boundaries, divided by ‘100’ and then multiplied by ‘10’ to the power of another randomly generated number, ranging between ‘1’ and ‘3’. Whereby, the magnitude and fidelity of the numbers generated, is resultant of the powering number passed to the divisor of each constant’s randomisation; to note, each constants randomisation is passed with a separately randomised power, in favour of increasing the yields of fidelity and variation for the current and proceeding processes of the algorithm. In which, the ranges configured for their randomisation aim to satisfy more variation, through the minimalization of biases which inversely increases comparability, when evaluating output of increased variation and magnitudinal difference between constant values; this purposes to hasten the rate of the tuning procedure, whilst enabling the potential output to be more optimal, for a larger array of unique output that is compiled. In knowing of the manual tuning test data, prior to the auto-tuning algorithm being configured, the minimum (‘0.001’) and maximum (‘100’) boundaries for their randomisation were appropriated in correspondence to the values tested; where ranges situated below or above the configuration presented, would not provide balance in the robots “performance and robustness” [15], which is relevant to its behavioural compliance and adequacy of rate in its adjustment to calculated error.

Beyond the proportionality constants being randomised, the state of a Boolean variable representing the randomisation of the constants, is alternated, to indicate the initiation of the iterative reasoning procedure for the algorithm. For said procedure, both the difference in RMSE value and count of oscillatory motions exhibited by the robot when within the edge following state, are calculated, and then used to condition and determine the adjustment process, for the currently cycled constant. In which, the RMSE calculations and variables configured in the manual tuning procedure prior, are recycled, meanwhile the previous RMSE value is stored as a separate float variable, and is set to the current RMSE value calculated, at the end of the frame being executed. Meanwhile, for determining the count of oscillatory motions exhibited by robot, when transitioning into edge following for either of its sides and whilst explicitly edge following, multiple integer variables are incremented for both scenarios where oscillations are anticipated.

When the robot invocates the subsidiary edge following transition state, oscillations are determined by every function call of the tuning implementation, when the constant variables have been initialised and the robot has not been following a wall, for a period of or more than ‘10’ seconds, which is represented as the accumulation of timestep, stored as a float variable. Given that the robot can endure continual changeover between the edge following transition, left and right subsidiary states, sizable oscillatory motions exhibited by the robot, can cause said transition to occur per oscillation performed; thereby, for every changeover to the edge following transition state, the related oscillation count is incremented, simultaneously to the edge following timer being reset. Moreover, for determining the number of sizable oscillations performed by the robot when explicitly edge following, a float variable representing the largest error calculated for the current span of uninterrupted edge following, is used to store the largest error calculated, so that the current error calculated for every frame, can be respectively compared. For the comparison declared, upon the currently calculated error surpassing the largest error recorded, the relevant oscillatory count increments simultaneously to the largest error updating to the value of the current; this strategy was deemed appropriate for measuring oscillation, given the function of the PID controller: to marginalise error and adjustment to the set-point overtime. Inevitably, larger errors contradict this purpose, which assumes that a larger adjustment to the set-point is required, hence sizeable oscillations are more likely to be actuated by the robot, in which they can be calculated and therefore recorded.

Advancing to their application as determiners of the adjustment process performed, upon the RMSE value or oscillation count increasing (worsening) in proportion to the adjustment that was issued for the affected constant prior, the algorithm inflicts an inversed adjustment to the constant or alternatively, the algorithm cycles through to the next iteration and potentially minimises the fidelity of change inflicted, before adjusting the next constant in the order of PID; otherwise, the algorithm continues to perform the same operation of adjustment, with a decreasing fidelity in change over time, for the same constant affected. For the process of adjustment mentioned, the operation performed is determined by the state of a Boolean variable, representing whether or not the adjustment operation is incrementing (true) or decrementing (false) the value for a given constant cycled; where each operation is processed as simple addition or subtraction from the constants current value. Given the binomial nature of the variable, the defaulted operation (increment) of the algorithm does not encourage the algorithm to behave differently to its counterpart state, when defaulted; this is achieved within the allowance configured for adjustment operations to alternate, for the same constant and cycle of the algorithm.

Meanwhile, in continuation of cycling, the algorithm recursively iterates the entirety of the reasoning procedure for a given set of randomised PID constants and for a total of one-hundred iterations; this was appropriated for enabling the fidelity in change issued to the constants, to depreciate over time, for the purpose of yielding higher degrees of optimisation. In which, said fidelity in adjustment reduces by a factor of ‘10’ for every ‘25’ cycles succeeded by the algorithm, which dictates the eventuality of constant change, to climax to four decimal places; to note, the adjusting amount is stored as a float variable, that is defaulted and reset to the value of ‘1’, for the initial twenty-five and one-hundredth cycles being succeeded by the algorithm.

For contextualizing the definition of a cycle in respect of the algorithm configured, cycling is represented as an integer variable which is a representative of the number of times that the PID constants have been adjusted and iterated over, collectively. Supporting this process is a separate integer variable, that exists to represent the index of the PID constant (in the array variable that they are all contained within) currently being tuned, and for cycling between the constant being tuned in the cyclical order of proportional (indexed ‘1’), integral (indexed ‘2’) and derivative (indexed ‘3’); this is all directed by the variable being incremented or reset, upon each constants adjustment operation being alternated, twice. This algorithmic structure enables each constants value to be incremented and decremented per iteration of the algorithm, allowing each constants value to be rectified for the compatibility and optimisation of the robot’s behaviours, that are exhibited whilst edge following. For invocating change within the value of the indexing variable, a subsequent integer variable exists to appropriate the number of times that the adjustment operation has alternated, for the constant currently being tuned; for every worsened adjustment, the value of the variable increments and upon alternating the adjustment operation twice, the indexing variable increments simultaneously to the alternation variable being reset. Proceeding from the incrementation of the indexing variable, upon said variable cycling to the derivative constant (indexed ‘3’) and the alternation variable climaxing to ‘2’ (either operation has been performed on the constants value), both variables are reset simultaneous to the algorithmic cycle incrementing; this signals the execution for the next iteration of the algorithm, where the process recurses with changes in adjustment fidelity over time. Upon the algorithm passing the one-hundredth cycle, all the variables associated with the algorithm are reset to their default configurations and a new set of constant values are randomised, for fine-tuning.

For every adjustment allotted to the constant’s values, the current configuration for the current iteration of the algorithm was output to an external comma-separated values (CSV) file, where in use of the Microsoft Excel software application, the data could be sorted to determine the most compatible vales, from a strictly numerical outlook. When sorting the data collected, higher precedence is offered to lower counts of sizeable oscillatory motions, for when the robot transitions into edge following behaviours; this decision was settled for prioritising collision avoidance, through which, the robot requires to exhibit fluidity in behavioural shift to uphold. Inevitably, sizable oscillations hinder said fluidity and therefore potentialize collision occurrence to be more frequent, than compared too few to no oscillations being exhibited. Thereby, as the secondary sorting arrangement, a lower count of sizable oscillatory motions exhibited by the robot whilst edge following, was also favoured more immediately in comparison to the resulting RMSE value compiled. For this filtering-based strategy, the likelihood of the robot performing to its expectations is bettered, where the “trade-off between performance and robustness” [15] is concluded in favour of behavioural consistency, as opposed to the robot’s performance in marginalising error quicker; where RMSE is calculated to represent.

IV. Results

1. *Manual fine-tuning*

For the test cases conducted manually from an observational standpoint, it was realized that the PID controllers’ parameters could be fine-tuned conventionally, with the relief of graph objects and console output supported by the IDE; in attempt to relegate the discrepancies between the expected and actual behaviours exhibited by the robot, when edge following. Whereby, in conclusion of the test results populated, the values that were discovered and configured for optimal behavioural exhibition, whilst within reasonable numerical fidelity for the time constraints posed, were: *proportional – 9.5*, *integral – 5.5* and *derivative – 0.25.*

In response to the configuration elected, the robot validates its ability to follow the edges of detected walls seamlessly, where it can support self-navigation around the environment that it is subjected to, whilst maintaining a constant detection with said walls, for preventing the invocation of random wandering behaviours and overcoming the perpendicular arrangements of the walls positioned in the environment. Conclusively, for the PID controller configuration discovered, the robot adheres to all its expectations tasked, for the environment that it is featured in.

1. *Autonomous fine-tuning*

In direct correspondence to the data that was handled and output to the datasheet setup, the values that were regarded by the algorithm as the optimal configuration, were: *proportional – 17.111*, *integral – 6.5418* and *derivative – 6.0526*. These values were determined by the sorting process detailed prior (‘125,472’ results considered), which favours the lower counts of oscillatory motions exhibited by the robot, when the edge following transition, left and right subsidiary states are active (robustness), as opposed to directly referring to the resultant RMSE value (0.007199) compiled (performance). This process as previously discussed, was appropriated for the robot’s expectation to achieve seamless behavioural-state transitioning, for nullifying collision occurrence(s); whereas when referring to the RMSE value immediately, the robot’s behaviours cannot be inferred for varied scenarios figured.

***Available in Appendix J***

When the configuration was regulated by the robot in the environment that it was exposed to originally, the robot demonstrates competency in navigating around the walls in the environment, exclusive to interruption; where the robot remains within the edge following state for the entirety of the simulation, upon a wall initially being detected. More so, the robot does not exhibit sizable oscillatory motions when aligning with a wall to follow and whilst following a wall that it is already aligned with.

Thereby, in conclusion of the autonomous fine-tuning conducted for the PID controller, the closing configuration compiled by the algorithm is compatible for addressing all the expectations projected, for the robot’s edge following behaviours. When compared to the test results obtained from the manual fine-tuning procedures, auto-tuning performed better in focus of the robot settling to the set-point faster; this is given by the resultant difference between the RMSE values compiled for both testing methods, of ‘0.0001’, for the equivalent environment. However, for each configurations adherence to the behaviours expected, the robot performs similarly, for either configuration applied.

V. Conclusion

In summary of the PID controller configuration submitted, for the Pioneer P3-DX mobile robot, it is certain that the robot can exhibit edge following behaviours proficiently, which is supported by the series of test cases conducted, that are varied by procedure to better the robot’s performance and behavioural compatibility, in the environment it was subjected to.

Throughout the implementation of the PID controller, my understanding of the fine-tuning processes involved in appropriating proportionality constants has developed, significantly, within the domains of both theoretical and practical applications. As has the approach to implementing an algorithm, for fine-tuning a PID controller, autonomously.

Given more time was available for the controller’s development, further adaptations would be made to the auto-tuning algorithm, for improving the performance of compatible configurations that it compiles; one suggestion would be to continue to experiment with increasing the numerical fidelities of the PID parameters passed.

References

[1] Génération Robots (2020) *Pioneer P3-DX mobile robot*. [Online] Génération Robots. Available from: https://www.generationrobots.com/en/402395-robot-mobile-pioneer-3-dx.html [Accessed: 24/11/20].

[2] KUMAR, D.U. and NISHA, N.M. and MATHIVANAN, N. (2018) Tracking of a PID Driven Differential Drive Mobile Robot. *International Journal of Mechatronics, Electrical and Computer Technology*. [Online] 8 (27). Available from: http://www.aeuso.org/index.php?volume=48&issue=46 [Accessed: 24/11/20].

[3] ESPINOSA, F. and SALAZAR, M. and PIZARRO, D. and VALDES, F. (2010) *Remote and Telerobotics*. London: IntechOpen.

[4] BIRARI, A. and KHARAT, A. and JOSHI, P. and PAKHARE, R. and DATAR, U. and KHOTRE, V. (2016) Velocity Control of Omni Drive Robot using PID Controller and Dual Feedback. In: *IEEE First International Conference on Control, Measurement and Instrumentation (CMI),* *Kolkata, India, 8-10th January 2016.* New York:IEEE, pp. 295-299.

[5] Control Soft Incorporation (2020) PID Tuning Objectives and Considerations. [Online] Control Soft Incorporation. Available from: https://www.controlsoftinc.com/pid-tuning-objectives-and-considerations/ [Accessed: 24/11/20].

[6] BARSAN, A. (2019) Position Control of a Mobile Robot through PID Controller. *Acta Universitatis Cibiniensis Technical Series*. [Online] 71 (1). Available from: https://www.researchgate.net/publication/338473586\_Position\_Control\_of\_a\_Mobile\_Robot\_through\_PID\_Controller [Accessed: 25/11/20].

[7] Coppelia Robotics (2020) *CoppeliaSim from the creators of V-REP*. [Online] Coppelia Robotics. Available from: https://www.coppeliarobotics.com [Accessed: 25/11/20].

[8] TOAL, D. and FLANAGAN, C. and JONES, C. and STRUNZ, B. (1995) Subsumption Architecture for the Control of Robots. [Online] Available from: https://www.researchgate.net/publication/244321030\_Subsumption\_Architecture\_for\_the\_Control\_of\_Robots [Accessed: 25/11/20].

[9] SIMPSON, J. and JCOBSEN, C.L. and JADUD, M.C. (2006) Mobile Robot Control the Subsumption Architecture and occam-pi. In: *The 29th Communicating Process Architectures Conference, CPA 2006*, organised under the auspices of WoTUG and the Napier University, Edinburgh. Berlin: ResearchGate, pp. 225-236.

[10] ANG, H.K. and CHONG, G. and LI, Y. (2005) PID Control System Analysis, Design, and Technology. In: *IEEE Transactions on Control Systems Technology*, *20th June 2005.* New York:IEEE, pp. 559-576.

[11] LEKKALA, K. and MITTAL, V.K. (2014) PID Controlled 2D Precision Robot. [Online] Available from: https://www.researchgate.net/publication/288424236\_PID\_controlled\_2D\_precision\_robot [Accessed: 25/11/20].

[12] KIM, K. and KIM, M. and CHONG, N.Y. (2010) RFID based collision-free robot docking in cluttered environment. In: *Progress in Electromagnetics Research.* Massachusetts: EWM Publishing, pp. 199-218.

[13] BANSAL, H.O. and SHARMA, R. and SHREERAMAN, P.R. (2012) PID Controller Tuning Techniques: A Review. *Journal of Control Engineering and Technology (JCET)*. [Online] Available from: https://www.researchgate.net/publication/316990192\_PID\_Controller\_Tuning\_Techniques\_A\_Review [Accessed: 26/11/20].

[14] HUSSAIN, K.M. and ZEPHERIN, A.R. and KUMAR, M.S. and KUMAR, S.M.G. (2014) Comparison of PID Controller Tuning Methods with Genetic Algorithm for FOPTD System. *Journal of Engineering Research and Applications*. [Online] 4 (2). Available from: https://www.semanticscholar.org/paper/Comparison-of-PID-Controller-Tuning-Methods-with-Hussain-Allwyn/8f8e0ba288f2a4b272ff8e7533ec1f1167ee7eda [Accessed: 26/11/20].

[15] HANI, H.A. and MASHOR, M.Y. and MAHDI, M.C. (2018) Performance of Manual and Auto-Tuning PID Controller for Unstable Plant-Nano Satellite Attitude Control System. In: *The 6th International Conference on Cyber and IT Service Management (CITSM 2018), Indonesia, 7-8th August 2018*. Berlin: ResearchGate, pp. 1-5.

[16] Agile Alliance (2020) *Exploratory testing*. [Online] Agile Alliance. Available from: https://www.agilealliance.org/glossary/exploratory-testing/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa\_book~'aa\_event\_session~'aa\_experience\_report~'aa\_glossary~'aa\_research\_paper~'aa\_video)~tags~(~'exploratory\*20testing))~searchTerm~'~sort~false~sortDirection~'asc~page~1) [Accessed: 26/11/20].

[17] MAJCHZRAK, J. and MICHALSKI, M. and WICZYNSKI, G. (2009) Distance Estimation with a Long-Range Ultrasonic Sensor System. *IEEE Sensors Journal*. [Online] 9 (7). Available from: https://www.researchgate.net/publication/224492926\_Distance\_Estimation\_With\_a\_Long-Range\_Ultrasonic\_Sensor\_System [Accessed: 26/11/20].

[18] CHAI, T. and DRAXLER, R.R. (2014) Root-mean-square Error (RMSE) or mean-absolute Error (MAE)? Geoscientific Model Development Discussions. [Online] 7 (1). Available from: https://www.researchgate.net/publication/262980567\_Root\_mean\_square\_error\_RMSE\_or\_mean\_absolute\_error\_MAE [Accessed: 26/11/20].

Appendices

***Appendix A:***



Figure : Mobile robot, the Pioneer P3-DX [1].

***Appendix B:***

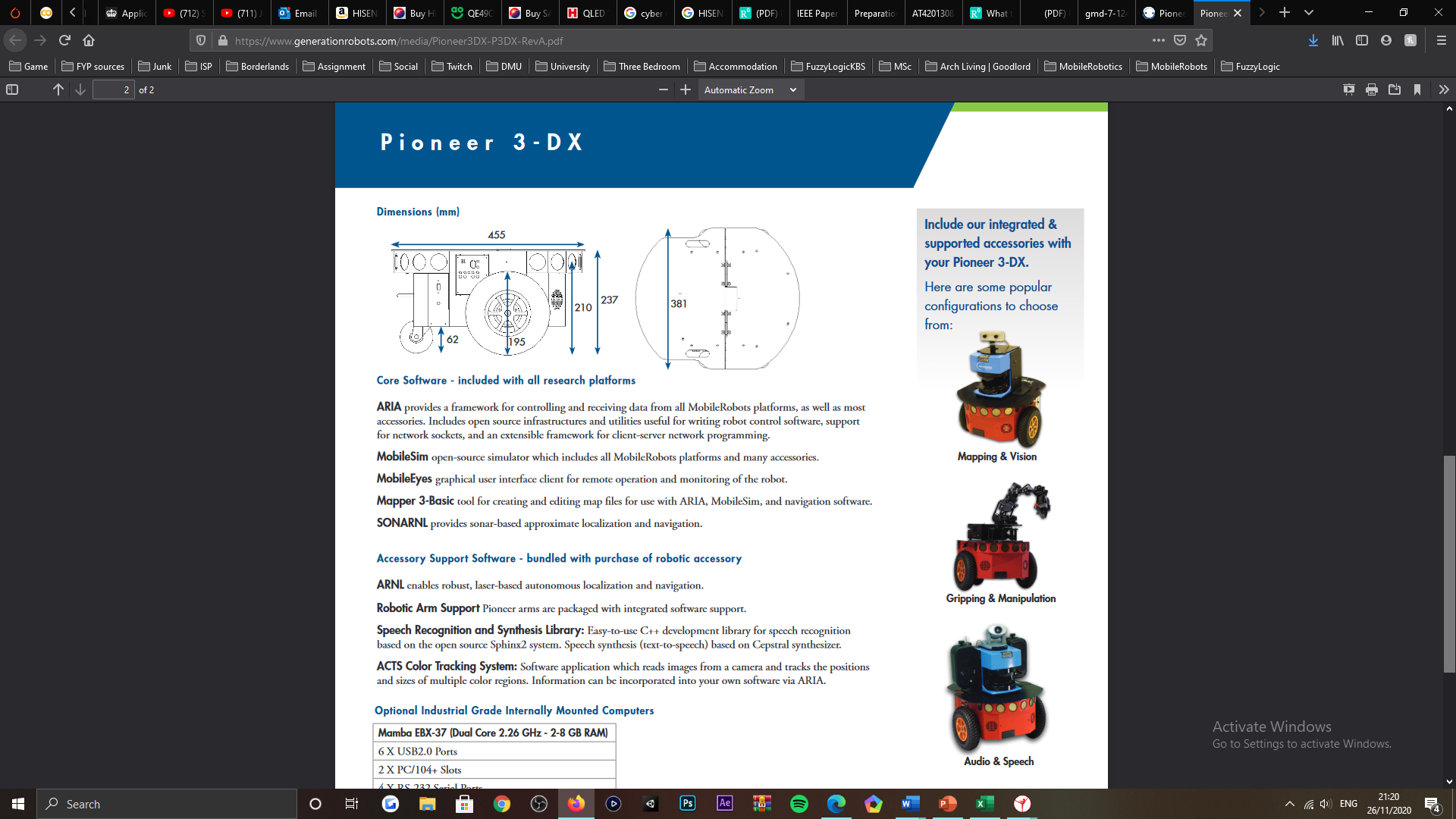


Figure : Visualisation of the wheel alignment for the Pioneer P3-DX mobile robot [1].

***Appendix C:***

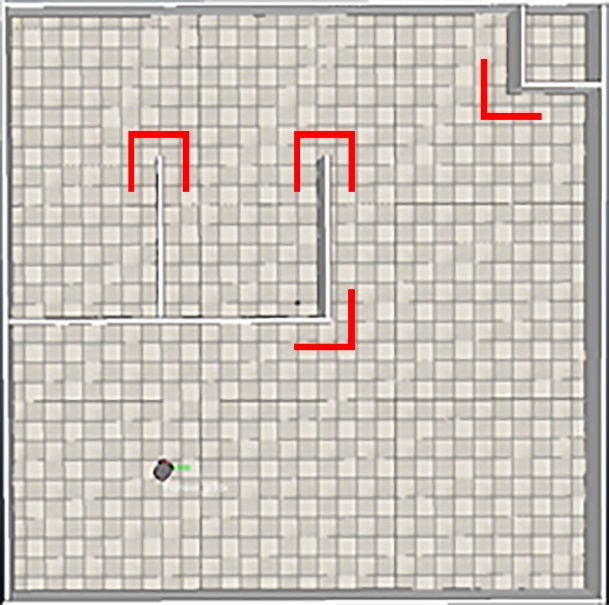


Figure : Wall objects that align with interior and exterior-angles for the robots subjected environment, this presents challenge to the robots ability in maintaining a detection with the faces of the walls shown, as well as remaining within the edge following state.

***Appendix D:***

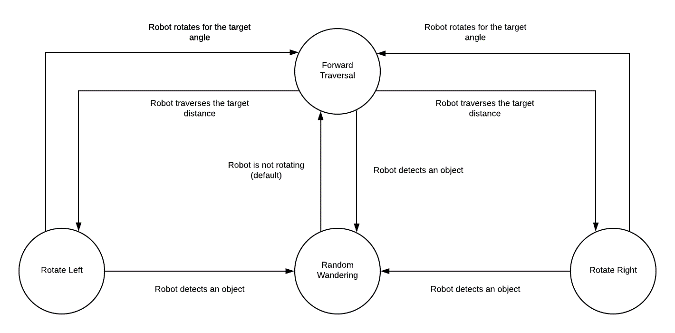


Figure : Finite-state machine (FSM), representing the behavioural flow between the major and subsidiary random wandering states.

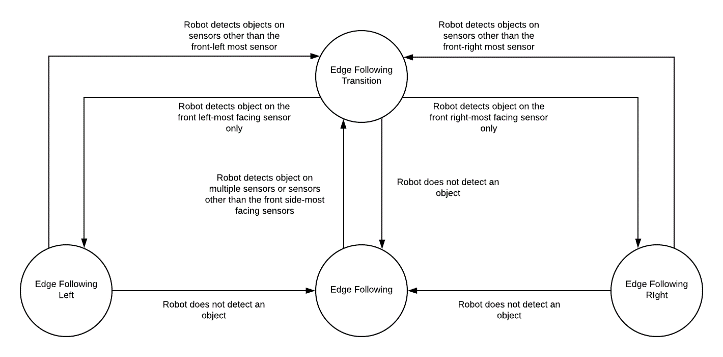


Figure : Finite-state machine (FSM), representing the behavioural flow between the major and subsidiary edge following states.

***Appendix E:***

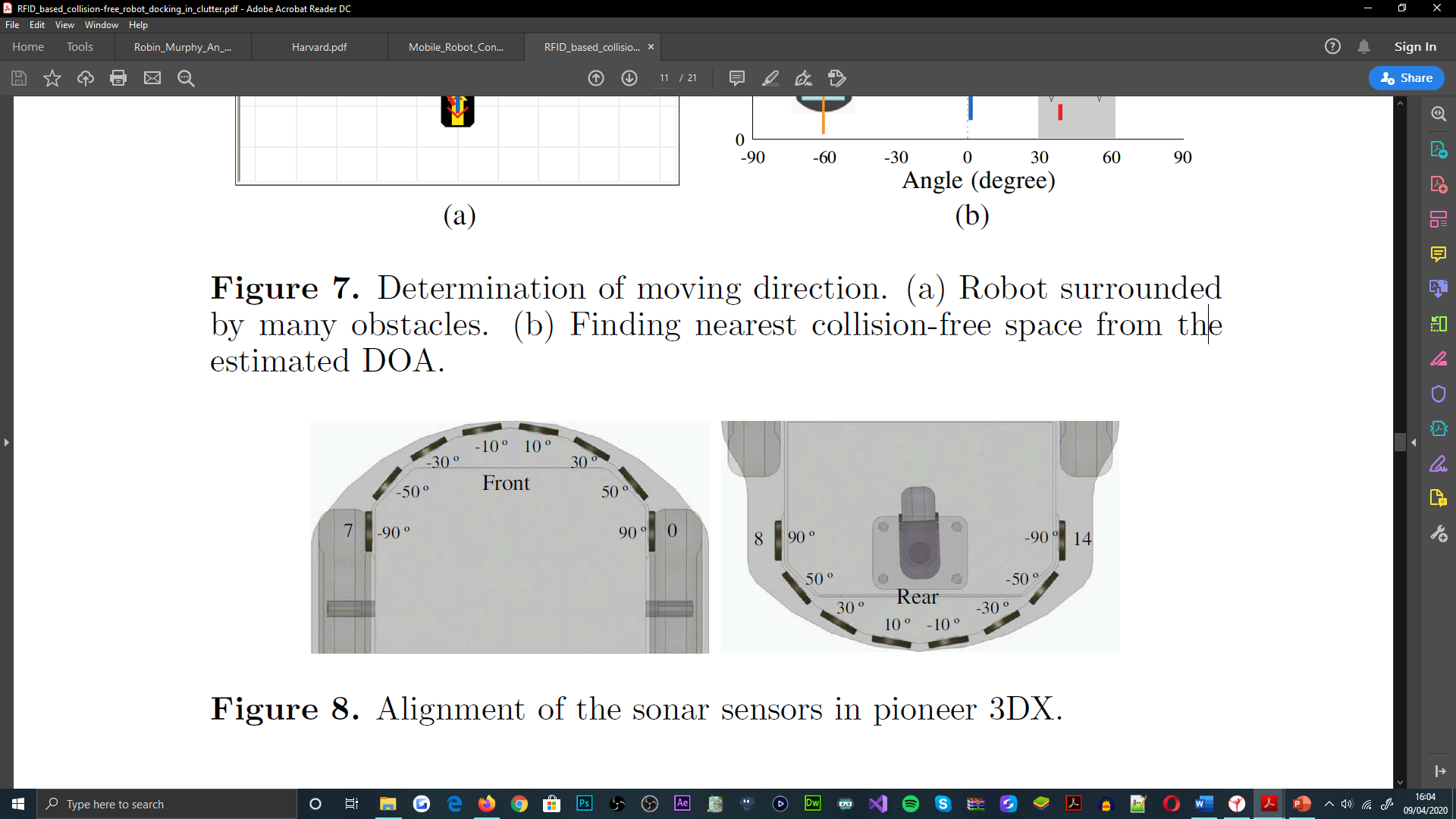


Figure : SONAR sensor positional and angular offsets, for the Pioneer P3-DX mobile robot [12].

***Appendix F:***

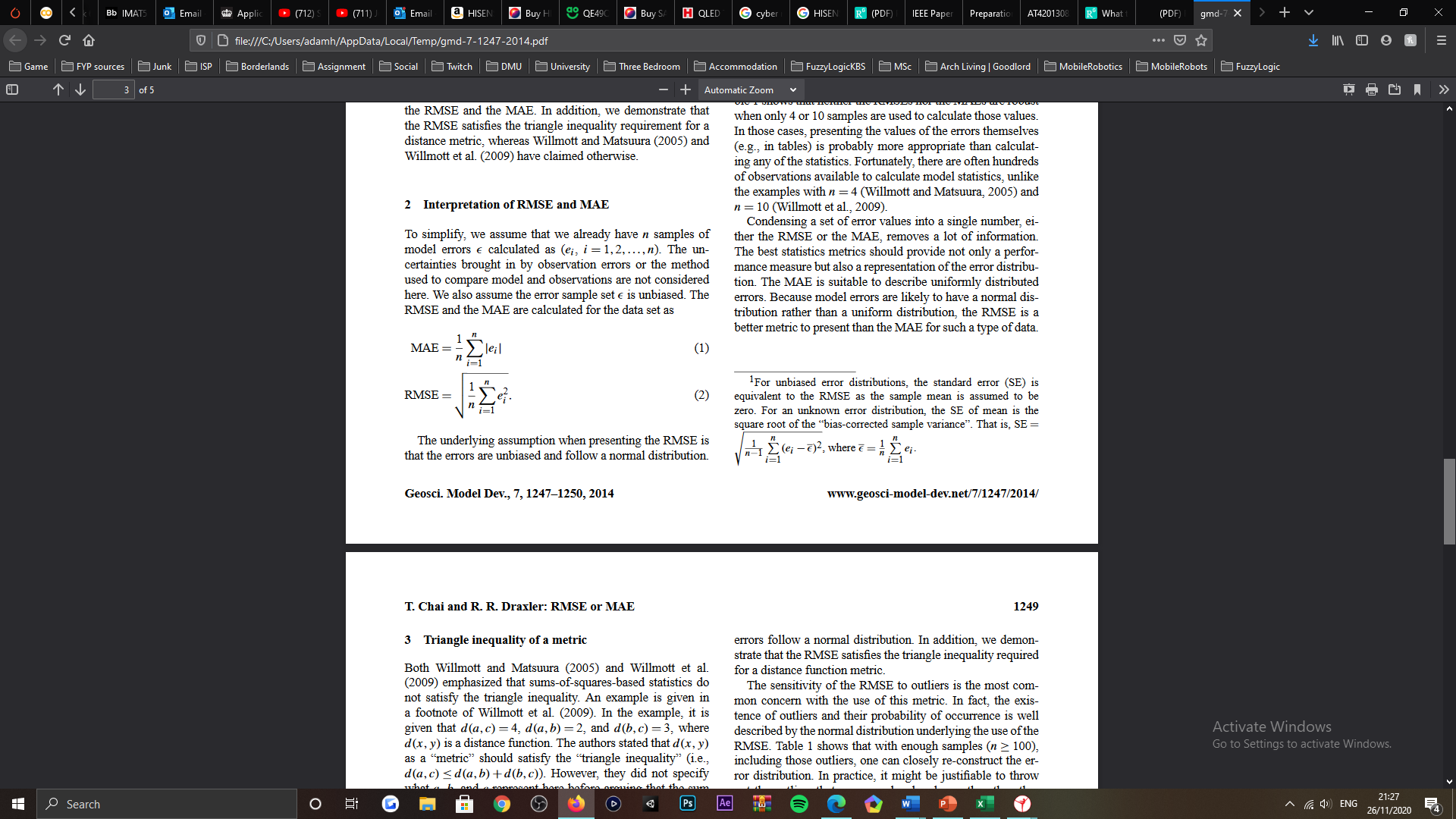


Figure : Root-mean-square error (RMSE) equation [18].

***Appendix G:***

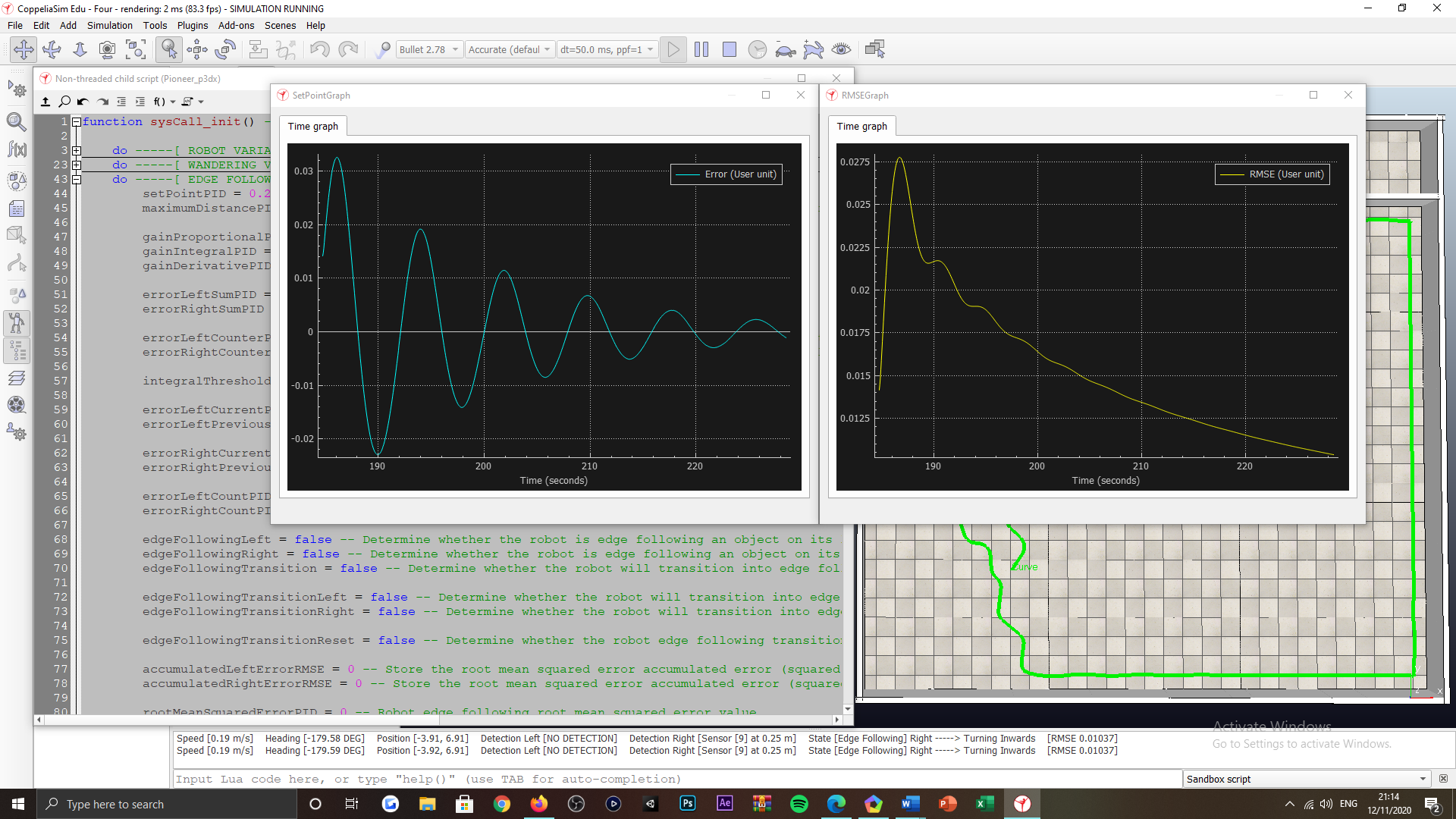


Figure : Proportional-integral-derivative (PID) controller error (left) and resulting PID root-mean-square error (RSME) (right) time graph plots, visualizing the marginalisation in error to the set-point, over time.

***Appendix H:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***PID controller proportionality constant values, for edge following behaviours*** | | | | | |
| ***Case*** | ***Constant*** | ***Value*** | ***Observation*** | ***RMSE*** | ***Final*** |
| ***1*** | Proportional | 1 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slow and small oscillatory motions exhibited by robot when maintaining set-point.* | 0.019406 | No |
| Integral | 0 |
| Derivative | 100 |
| ***2*** | Proportional | 1 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many faster and larger oscillatory motions exhibited by robot when maintaining set-point.* | 0.022312 | No |
| Integral | 0 |
| Derivative | 200 |
| ***3*** | Proportional | 1 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.0194062 | No |
| Integral | 0 |
| Derivative | 50 |
| ***4*** | Proportional | 40 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many faster and larger oscillatory motions exhibited by robot when maintaining set-point.* | 0.002990 | No |
| Integral | 0 |
| Derivative | 50 |
| ***5*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.003651 | No |
| Integral | 0 |
| Derivative | 50 |
| ***6*** | Proportional | 19 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.008509 | No |
| Integral | 0 |
| Derivative | 50 |
| ***7*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.003599 | No |
| Integral | 0.001 |
| Derivative | 50 |
| ***8*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.003595 | No |
| Integral | 0.01 |
| Derivative | 50 |
| ***9*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.003708 | No |
| Integral | 0.1 |
| Derivative | 50 |
| ***10*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.003597 | No |
| Integral | 1 |
| Derivative | 50 |
| ***11*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many faster and larger oscillatory motions exhibited by robot when maintaining set-point.* | 0.002848 | No |
| Integral | 10 |
| Derivative | 50 |
| ***12*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.003199 | No |
| Integral | 5 |
| Derivative | 50 |
| ***13*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many faster and larger oscillatory motions exhibited by robot when maintaining set-point.* | 0.003179 | No |
| Integral | 7.5 |
| Derivative | 50 |
| ***14*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.003220 | No |
| Integral | 6 |
| Derivative | 50 |
| ***15*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.003220 | No |
| Integral | 6 |
| Derivative | 25 |
| ***16*** | Proportional | 20 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.003220 | No |
| Integral | 6 |
| Derivative | 20 |
| ***17*** | Proportional | 15 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.003953 | No |
| Integral | 6 |
| Derivative | 20 |
| ***18*** | Proportional | 15 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.003953 | No |
| Integral | 6 |
| Derivative | 10 |
| ***19*** | Proportional | 15 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.003953 | No |
| Integral | 6 |
| Derivative | 5 |
| ***20*** | Proportional | 15 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.003953 | No |
| Integral | 6 |
| Derivative | 2.5 |
| ***21*** | Proportional | 15 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.004304 | No |
| Integral | 6 |
| Derivative | 1 |
| ***22*** | Proportional | 15 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.004304 | No |
| Integral | 6 |
| Derivative | 0.1 |
| ***23*** | Proportional | 10 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.006107 | No |
| Integral | 6 |
| Derivative | 0.1 |
| ***24*** | Proportional | 5 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower yet larger oscillatory motions exhibited by robot when maintaining set-point.* | 0.010611 | No |
| Integral | 6 |
| Derivative | 0.1 |
| ***25*** | Proportional | 7.5 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many faster yet smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.009342 | No |
| Integral | 6 |
| Derivative | 0.1 |
| ***26*** | Proportional | 8 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.007622 | No |
| Integral | 6 |
| Derivative | 0.1 |
| ***27*** | Proportional | 9 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.006177 | No |
| Integral | 6 |
| Derivative | 0.1 |
| ***28*** | Proportional | 9.5 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower yet larger oscillatory motions exhibited by robot when maintaining set-point.* | 0.006959 | No |
| Integral | 6 |
| Derivative | 0.1 |
| ***29*** | Proportional | 9 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower yet larger oscillatory motions exhibited by robot when maintaining set-point.* | 0.008990 | No |
| Integral | 3 |
| Derivative | 0.1 |
| ***30*** | Proportional | 9 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.008558 | Yes |
| Integral | 4 |
| Derivative | 0.1 |
| ***31*** | Proportional | 9 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many faster yet smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.007250 | No |
| Integral | 5 |
| Derivative | 0.1 |
| ***32*** | Proportional | 9 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.007213 | No |
| Integral | 5.5 |
| Derivative | 0.1 |
| ***33*** | Proportional | 9 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.007213 | No |
| Integral | 5.5 |
| Derivative | 0.05 |
| ***34*** | Proportional | 9.5 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.007213 | No |
| Integral | 5.5 |
| Derivative | 0.2 |
| ***35*** | Proportional | 9.5 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many faster and larger oscillatory motions exhibited by robot when maintaining set-point.* | 0.007213 | No |
| Integral | 5.5 |
| Derivative | 0.5 |
| ***36*** | Proportional | 9.5 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Many slower and smaller oscillatory motions exhibited by robot when maintaining set-point.* | 0.007213 |  |
| Integral | 5.5 |
| Derivative | 0.3 |
| ***37*** | Proportional | 9.5 | *Robot maintains detection with all arrangements of walls and adjusts to set-point over time. Similar frequency and magnitude of oscillatory motions exhibited by robot when maintaining set-point.* | 0.007213 | Yes |
| Integral | 5.5 |
| Derivative | 0.25 |

Table : PID controller, proportionality constant value test cases, used to govern the behaviours exhibited by the robot when within the edge following state.

***Appendix I:***

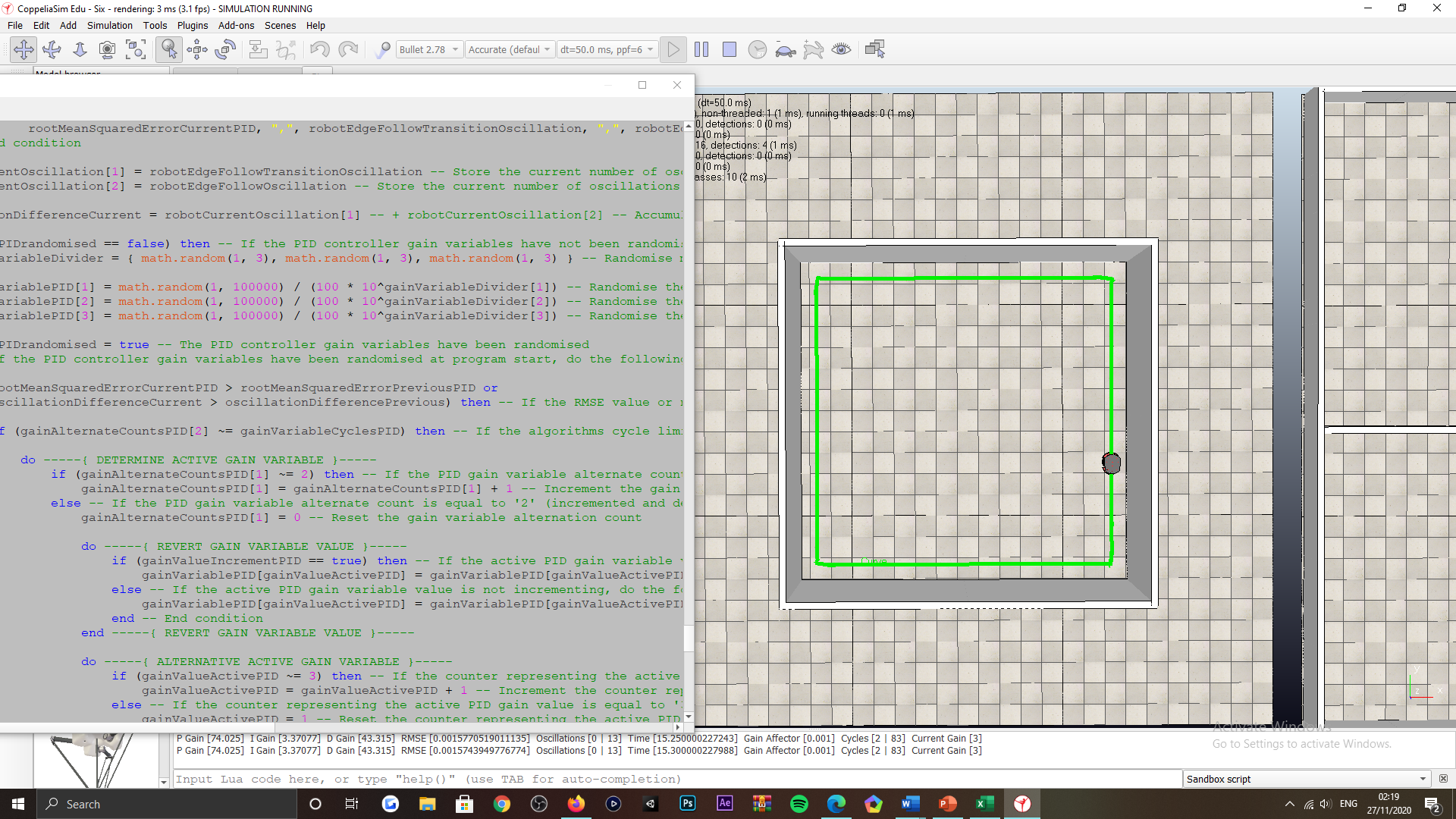


Figure : Visualisation of the enclosed environment exposed to the robot, for the auto-tuning procedures of the PID controller’s proportionality constants.

***Appendix J:***

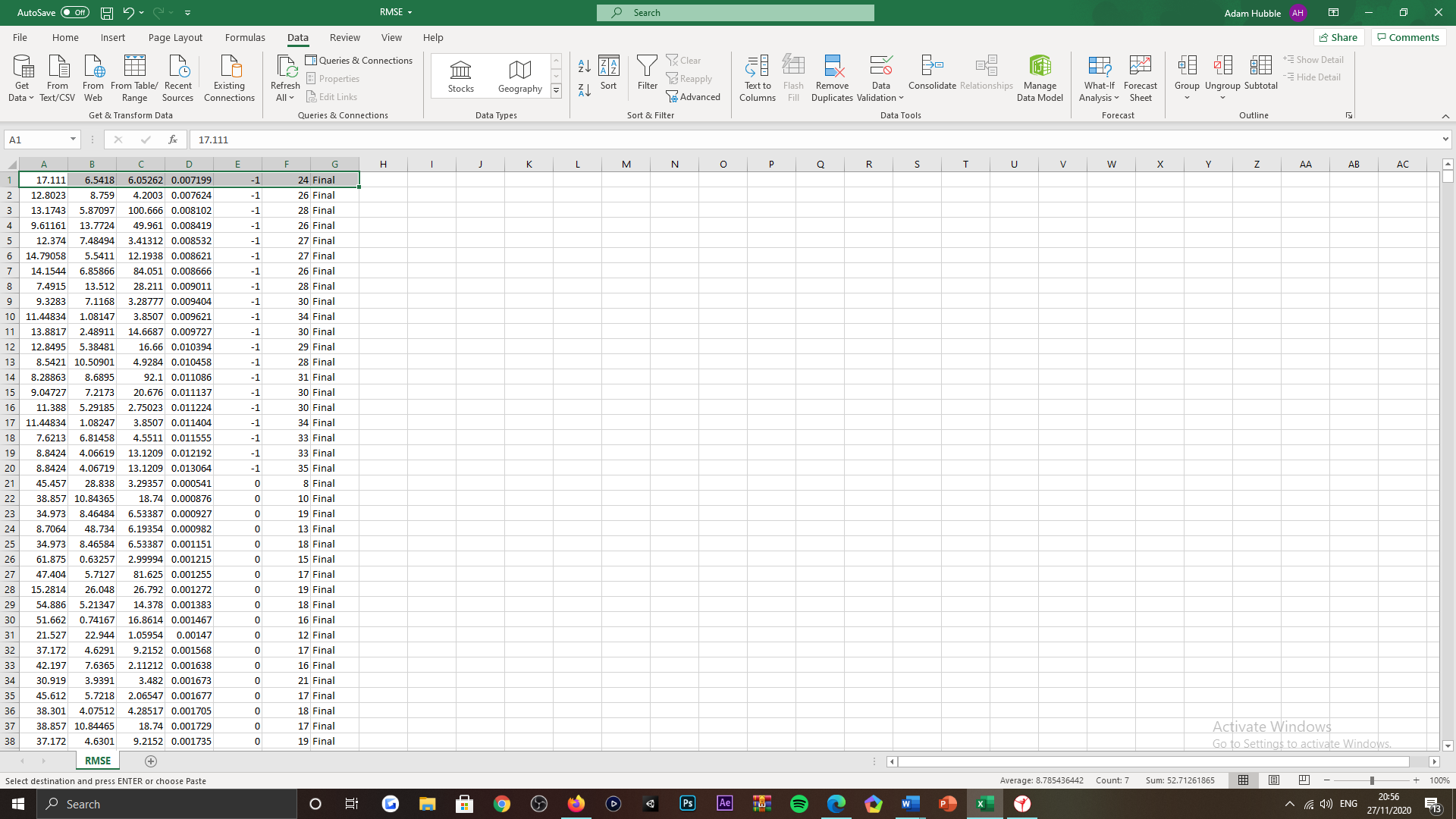


Figure : Autonomous tuning output, datasheet populated with PID proportionality constant values, RMSE values, sizable oscillations exhibited by the robot when transitioning into the edge following state and sizeable oscillations when explicitly edge following (left to right).