Integrating Optimized High-Speed Autonomous Control Systems

James V. Johnson, Alfred DeGennaro, Kenneth Gregson

South Carolina Governor's School for Science and Mathematics, Massachusetts Institute of Technology Lincoln Laboratories

While autonomous vehicles are growing in popularity, there exists a tradeoff between control and speed. With control directly affecting safety of a vehicle, it is prioritized at the detriment of speed. However, with speed being essential to emergency responses, methods are required for optimizing speed while retaining high accuracy. This research project aims to convert a proportional integral derivative controller, visual servoing system, and potential field navigator into Robot Operating System (ROS) nodes, integrate them with the Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot (RACECAR) developed at the Massachusetts Institute of Technology, tune them for speed, and integrate them together to complete a test course of various obstacles. Our optimized algorithm variants achieved an increase in processing speed by 20 times, but did not outcompete completely rewritten variants in the test courses. This indicates that the optimization methods provide benefits, but that algorithms themselves must be rewritten to operate at the highest efficiency. Future research will incorporate the optimization methods into the rewritten algorithms to benefit from both effects. Additional algorithms will also be considered for revision and optimization.

Introduction

Interest in commercial autonomous vehicles has grown significantly in recent years. Tesla autopilot and the Google-splinter Waymo, two commercial success, exemplify the consumer-based autonomous cars currently produced. These autonomous vehicles focus on safety and reliability rather than speed or control. This choice makes the vehicles these companies produce non-optimal for emergency situations.

Speed, among other features like control and durability, remains an important characteristic of emergency and military vehicles. By applying autonomous algorithms to these characteristics, human drivers experience less stress during navigation. The Defense Advanced Research Projects Agency (DARPA) held the DARPA Grand prix in 2004 to specifically address this issue, ending with no completions. The same challenge was held in 2005 with 5 winners. Focusing on a changing environment, the DARPA Urban Challenge was also held in 2007, spurring even more creativity. While these results were encouraging, the winner only reached around 6.26 meters per second (14mph). Other projects have inspected these control systems, yet produce results too slow to consider[1], or focus on non-changing environments[2].

Figure 1 shows the Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot (RACECAR), developed at the Massachusetts Institute of Technology (MIT) to aid in development of autonomous algorithms[3]. The RACECAR is a one tenth scale model car outfitted with a 256-core processor (NVidia Jetson TX2), laser range finder (Hokuyo LiDAR), 9 degree-of-freedom inertial measurement unit (Sparkfun 9-dof IMU), and stereo camera (ZED). Using the Robot Operating System (ROS) Kinetic Kame on Ubuntu 16.04, the RACECAR can be programmed in Python, allowing for the development of various control systems[4]. ROS allows for an easily understood, quick, parallel-processing system on the RACECAR. It also inherently integrates the Python programming language, which allows the use of many integrated libraries. Examples include the default Python Math library, Open Computer Vision 2, and Numpy, all of which extend the Python language with various additional functions. The default Math library allows for highly optimized trigonometric functions. Open Computer Vision 2 (CV 2) provides image processing algorithms, such as contour detection, color recognition, and Hue Saturation Value interpretation[5]. Numpy provides various optimized algorithms which deal with raw arrays as well as gaussian, normal, and dot functions[6].

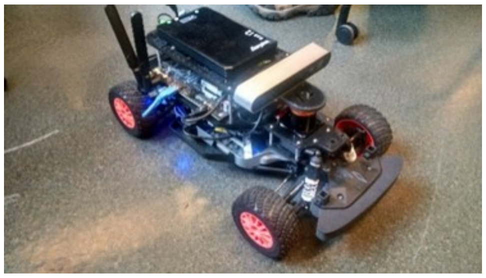

Figure 1: The RACECAR Platform

The speed of an autonomous vehicle depends heavily on the efficiency of the underlying algorithm. Thus, the project focused on four main algorithms: Proportional – Integral – Derivative control (PID), Color – based Line following, Potential Field exploration, and Machine State managing. Each algorithm provides useful characteristics which makes it ideal in a high-speed scenario. However, the algorithms also have drawbacks which may reduce speed, safety, or both depending on the situation.

PID provided a simple, yet easily optimizable function to achieve a goal – e.g. A velocity of distance – with little overshooting or inaccuracy. The algorithm focuses on preventing any extremely dangerous movement from a desired path, allowing for stability and quick recoveries. Unfortunately, the algorithm remains too simple. A PID system can only control one variable at a time, and requires another, separate algorithm for another controlled variable (K. Edelberg, lecture, July 12, 2017).

Color-based line following provided a simple identification of non-physical borders, allowing the RACECAR to remain within traffic lines or on painted paths. This system provides flexibility, utilizing only what the camera sees to navigate an area. The complexity of lighting in an environment, however produces a difficult tuning process, and differing environments require completely different set of tuning parameters due to unique lighting[7].

Potential field navigation allowed the RACECAR to remain heavily adaptable while guiding the RACECAR to empty space. This adaptability offers correction early enough to remain at high speeds in well-marked areas, while allowing for exploration to reorient erratic systems. The algorithm itself, however, reacts unpredictably, and can react too violently in extreme cases (Arslan, lecture, 26 July, 2017).

Finally, machine state managing allows for the use of multiple of the above algorithms, allowing the RACECAR to utilize algorithms in their best cases. Unfortunately, this system requires a unique trigger, and overwhelms a processor with the computation of each algorithm simultaneously (K. Edelberg, lecture, 12 July, 2017).

Ideally, a High-speed Autonomous vehicle could utilize versions of these algorithms to navigate throughout multiple environments, using a state manager to switch to the most suited algorithms for each environment. The algorithms' simplicity allows for quick development and testing for specialized environments; however, they require high levels of optimization.

Methods

The research group focused primarily on the optimization and specialization of each of these algorithms: Proportional Integral Derivative, Visual Servoing, Potential Field, and State Managing. The groups first addressed the specific environment that an algorithm performed the best in by assessing its strengths and weaknesses. Then, to optimize said algorithms, the group split into multiple teams focused on creating unique, efficient versions. Each team's result then participated in a race on a track modeled after the desired. This process repeated for each of the selected algorithms, and afterwards the teams joined back together to analyze the best performances for each race.

The group first focused on the Proportional Integral Derivative control algorithm. The PID's main strength arose from its simplicity, requiring only three calculations per cycle. Tuning, additionally, only consisted of three preset values. The teams experimentally determined these values according to their implementations of the algorithm. Trouble arose when scaled to higher speeds, however: the three variables used for tuning required immense precision to attain the desired state; otherwise the car oscillated off course and failed. When already at the desired state, the PID maintained accuracy. Thus, the group determined that the PID produced the best result on straightaways, when the RACECAR needed little adjustment to maintain course.

The group then focused on Visual Servoing. This algorithm relied only on the camera, which gave the RACECAR the ability to navigate in areas without guiding walls. The algorithm did not create as a simple a tuning process as the PID, but provided five variables to account for. The first three pertained to which color the algorithm would look out for, which the group determined beforehand. The other two controlled the physical attributes of the car – steering and speed. The teams experimentally determined these values. When scaling to higher speeds, the visual servoing algorithm did not have the computational power to correct in time. However, it still allowed for the traversal of open areas by marking on the floor, and thus the research group focused on specializing it for such areas.

Finally, the group analyzed the Potential Field algorithm. The Potential Field provided an adaptable navigation method. By using the laser range finder to scan the immediate area, the Potential Field analyzed everything in its field of view each cycle. The algorithm used 3 variables in tuning, with two focusing on how obstacles affect steering and speed, and a third adjusting base speed. The resultant Potential field algorithm performed exceptionally well in tight spaces with sharp turns. In open areas, however, the algorithm would waste time by driving at a slant, as it attempted to navigate away from any obstacle. Due to this, the research group determined it would operate best in enclosed, winding environments.

Once the group had assigned each algorithm a category, the group split into many teams to create separate implementations of each algorithm and specialize it to the desired environment. The research team focused on specific characteristics of the algorithms to optimize and revise. The team tuned the PID to have less influence from turning forces, and instead focused on its ability to maintain a straight path along the course. The team tuned the visual servoing algorithm to disregard unneeded data to reduce computation time. For the Potential field, the team created a lookup table for sine and cosine values to prevent repetitive computation of identical values. Finally, the team optimized the state manager to prevent noninfluential processes from taking up resources, allowing the main algorithm to run as quickly as possible.

The research team first analyzed the PID algorithm. Due to the group's decision to focus on straightaways, the team separated the algorithm into its three modalities: proportional, integral, and derivative. The proportional modality did not affect the maintenance of the straight line much, nor did the integral modality. The derivative modality did, however, affect the RACECAR's precision on course significantly, and the team focused on it.

The team's modified algorithm then competed in a drag race. Figure 2 shows the race track setup. Two RACECARs lined up at the start for each race, and then began after a countdown. The group disqualified any car which ran into a wall or another car. At the end of the race, a RACECAR must stop within the 2 meters between the finish line and back wall, or else the group disqualified it. The group split the teams into a 9-man bracket, with the winner representing the fastest and safest algorithm.

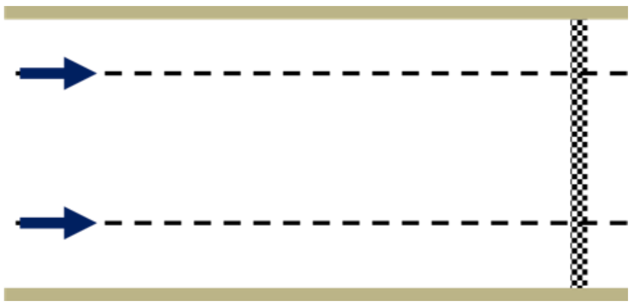

Figure 2: The Drag Race used to test PID algorithms

The team then analyzed the visual servoing algorithm. To specialize the algorithm to open areas with demarcations on the ground, the team focused mainly on the visual input data from the stereo camera to the car. The large amount of data sent to the algorithm slowed it down significantly, translating into slower reaction time for the RACECAR. After experimenting with stopping the program after finding relevant data, the team addressed this by simply cutting the data in half; the top half of the images from the camera contained only aerial features, which the algorithm did not need. This resulted in the algorithm having to operate on the half of the data which it needed. Additionally, the team added in a speed reduction when turning to allow the RACECAR to obtain accurate images of each new heading before it had traversed too far away from the line.

This algorithm competed against other team's algorithms in a time-trial consisting of various colored lines to follow. Figure 3 shows the general paths which a car could have followed, and the color separations. Each team had three trials to run the course in. Two of those trials had to follow the main orange line; the final trial could follow any of the lines. The group penalized the completion time of any RACECAR which drove off the line. Winners represented the algorithms which could turn the most accurately and quickly, getting to the finish line in the shortest times.

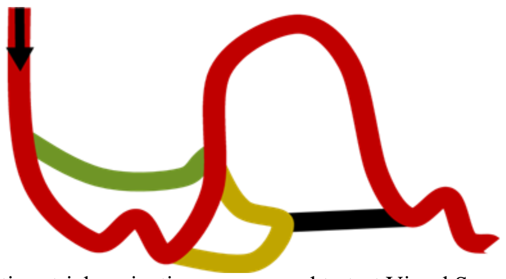
Figure 3: The time-trial navigation course used to test Visual Servoing algorithms

Afterwards, the team approached the potential field algorithm. With an emphasis on handling in tight spaces and corridors, the potential field could reach high speeds and still maintain precise control of the RACECAR. However, as speed increased, the reaction time of the program became more of a hindrance. At the desired speeds, the algorithm crashed into walls, and could not operate quickly enough for how fast it went.

To increase this reaction time, the group identified the most computationally expensive portions of the potential field: trigonometric functions. The functions transformed the polar coordinates of the laser range finder into cartesian coordinates, which then affected the horizontal and vertical aspects of the RACECAR: steering and speed respectively. By realizing the laser range finder gave the same angles for the polar coordinates, however, the team pre-computed each trigonometric function for the desired angles. The team then placed these values into a lookup table, which the algorithm referenced every loop instead of computing.

The group then tested each team's potential field algorithm on a winding course race. Figure 4 shows the general path of the course, which two cars raced down for every trial. The group placed two obstacles in the first and last turn in the relative middle of the track. Additionally, the track widened and narrowed at certain points to ensure the RACECARs could navigate changing environments. A bracket like that of the PID races organized which teams raced against each other. Teams earned penalties if they crashed into the other teams' RACECARs or the obstacles.

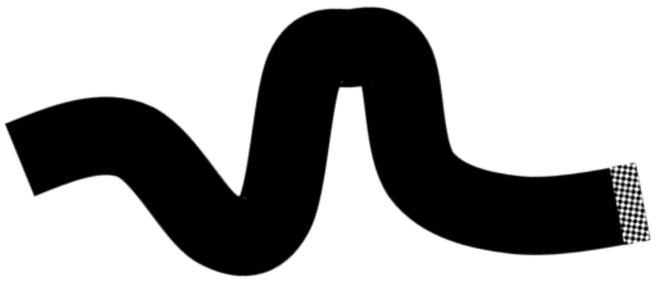
Figure 4: The winding track used to test potential field algorithms

Finally, the team focused on the state manager. By combining each of the above algorithms, the state manager possessed the strengths of each algorithm for its specified areas. However, the complexity this ensured presented the main challenge. Initially, the state manager did not live up to its task; in running each of the algorithms, the reaction time slowed down, preventing any of the algorithms from operating well at high speeds. To improve this behavior, the team prevented algorithms from running when not in use and decoupled the manager from its associated vision-based state identifier. This allowed the currently running algorithm access to all possible computational resources, granting it near the same reaction time as when run individually.

The teams then tested their state managers in the mini grand prix race. Figure 5 shows the full race, with each portion designed to test a certain attribute of the state manager. The first portion, the Rolling Weave, required the RACECAR to navigate a winding track. The track then transitioned into the Hairpin turn, a full 180-degree path minimizing the RACECAR's turning radius to test the controller's handling. The track then lead into the Overpass, a small elevated bridge providing both a straightaway and ramp for the cars to speed along. The Overpass ended in the Yellow Brick Road, a completely unwalled area demarked only by large yellow strip on the floor. Afterwards came the Underpass and Boa Basin, areas which provided two straightaways separated by a sharp turn. Finally, the Mesh wall led into the finish line, lined with a mesh designed to provide the laser range finder with minimal data to work with.

The teams first raced according to time in three individual trials. Any touching of the blue in the water hazard multiplied time by 1.1, 1.25, 1.5, or 2 depending on the number of wheels in the water. Crashing into the course walls at high speeds also penalized time, and any human intervention to fix the RACECAR added thirty seconds.

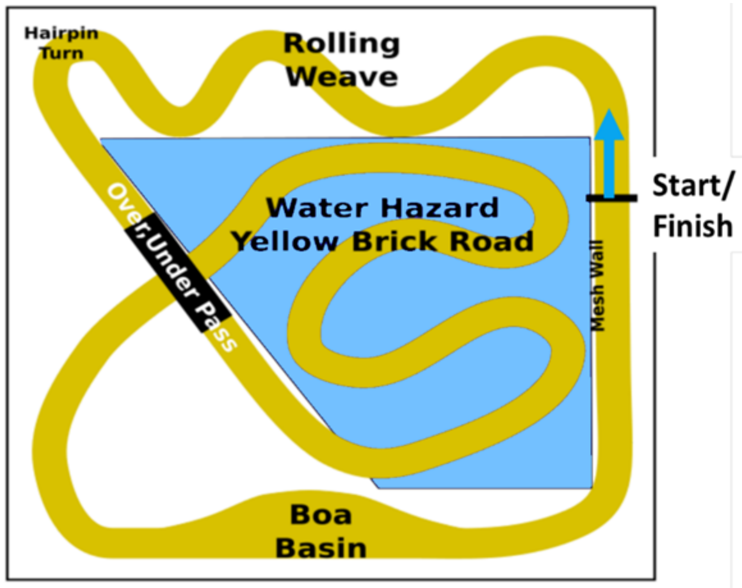

Figure 5: The mini grand prix race used to test all attributes of the state manager

Next, the teams separated into three groups of three and two groups of two for the heat races. Each grouping raced simultaneously to determine spots in the final full prix. Similar rules as the time trials applied, preventing cars from breaking rules to get to the start as soon as possible.

The final prix placed each group a RACECAR length away from each other, with horizontal spacing chosen according to placement in the heat races. The same rules applied as in each other race, with the upper places representing the quickest, most efficient algorithms.

Results

The research group analyzed each team's algorithms for each race by the competition winners. The teams also analyzed their own designs, finding the most characteristic features of their algorithms and determining how those affected their results

The team's PID controller achieved third place of nine teams, defeated only once by the competition winners. The RACECAR drove at a set speed around 3.5 meters per second, and stopped in the 2-meter braking area all but one time. Afterwards, the team analyzed how each of the three modalities preformed for the specified purpose.

The proportional modality did not participate much in the maintenance on a straightaway. Focused mainly on correction to desired state, the proportional force did little when the RACECAR already drove at the desired state. In lightly curving sections, the proportional could help to keep the car aligned with the wall; however, the other modalities could also aid this.

The integral modality also did little in the straightaway. Like the proportional, the integral force mainly allows the RACECAR to correct when off the desired path. When already there, it participated little in maintaining the path. It did, however, aid when at higher speeds. As the car travelled more distance, slight steering error would cause more of an effect. Therefore, the integral force would sum to much greater affect, and serve to adjust the slow error accumulation of the RACECAR.

The derivative modality did the most work in maintaining the path on a straightaway. By preventing the RACECAR from having any positive or negative angle from the wall, the derivative force would keep the RACECAR moving along the wall. The modality provided the drive stability, and because it directed all movement along the wall, it also aided speed. The team maximized this value making it heavily influential on the RACECR's movement.

When attempting to stop the visual servoing algorithm after finding the desired data, the RACECAR exhibited random behavior, due to the high possibility of recognizing erroneous colors as the line. Additionally, the algorithm still exhibited slow reaction times, as it had to operate through the entire image. However, when the algorithm only had to operate on the bottom half of the image, reaction time increased significantly. Due to this, the RACECAR made most of the earlier tight turns in the track.

In the competition, the team's visual servoing algorithm placed fourth in nine teams. The algorithm did exceptionally well on long, lightly curving areas, maintaining a high speed and remaining on the line. Additionally, the slow down on turns allowed the RACECAR to maintain precise headings on the track and navigate through the tight turns at the end of the course. This slow down, however, did reduce the RACECAR's speed enough that it did not complete the track as quickly as other team's cars.

The potential field algorithm's lookup table allowed the RACECAR to successfully navigate tighter areas at higher speeds than the original could manage. In the races, it claimed fifth place out of nine. The most trouble arose with the obstacles in the way of the path, especially in the first turn. By focusing on slowing down rather than turning randomly, the car lost large portions of time compared to the other teams. This led to a much slower algorithm than the other teams, despite the accuracy it provided.

The team's modified state manager increased the reaction speed significantly. Originally at a cycle rate of 7 Hz, the modified version ran at 140 Hz, increasing by a factor of 20. This produced a positive navigational effect, allowing the car intense reaction time.

In the individual time trials, the team's state manager placed seventh out of thirteen teams. In the rolling weave, the RACECAR operated at a moderate speed, allowing for the harsh transition into the Hairpin turn. The RACECAR took the turn itself well, taking a path close to the inside. The transition into the overpass slowed down the car due to the incline, but very quickly picked up after approaching. The transition into the Yellow Brick Road also slowed down the car, and occasionally the algorithm would not start up quickly enough to save the car from touching the blue; navigation also slowed the RACECAR until the Underpass, where it sped up. This transition also slowed the car down, and in testing would sometimes strand the car under the bridge. However, once on the track, the car would go significantly faster, and sped into the Boa Basin. The original strategy of switching from left wall to right failed multiple times, but staying on the left wall proved incredibly effective. Finally, the RACECAR performed exceptionally well on the Mesh Wall, reaching its highest speeds there.

The team achieved second place in the third Heat group, and placed on the left in the final race. Unfortunately, another team's RACECAR crashed into the team's car and knocked the power button, turning it off at the starting line. The race could not run again due to time constraints.

Discussion

The algorithms experienced a significant increase in computational speed, yet did not achieve a faster physical speed. Our algorithms did not reach the speeds of the rewritten versions, illustrating how merely optimizing the navigation algorithms did not maximize possible safe speed. Other rewritten systems did fall behind the optimized algorithms, however; one even ran into our RACECAR at the beginning of the match, disabling both cars. While this rendered the final race inconclusive, it proved that our algorithms remain a safe, viable option to compare against.

The navigation algorithms could be improved significantly, specifically by optimizing the algorithms more. Due to the time constraint, gains were chosen when they were 'good enough,' and they could certainly be improved upon. Specifically, a few gains in the algorithms could have used more inspection. The PID wall follower was far too unstable for its designated areas, and accordingly could have had the proportional modality lowered to compensate. Similar problems existed for the line follower, and it likely could have been made far less unstable with a similar change. Finally, the potential field algorithm was only useful in the rolling weave section of the track, and might have been more adaptable had it been able to take on straighter sections of track.

However, these changes clearly would not have allowed the car to gain first place. As we optimized the algorithms, the increase in algorithmic efficiency would asymptotically increase speed until reaching a base value for an algorithm. At that point, the team would need to rewrite the algorithm to increase speed.

The PID was far too unstable and unresponsive, requiring a certain distance from a wall to travel at maximum speed down the track. Additionally, it could not detect other cars effectively, making it possible for the test vehicle to follow a rival car should the rival car come between the wall and LiDAR sensor. The entire algorithm did not provide an improvement over using the potential field algorithm, which could have allowed for more dynamic navigation.

The line follower program also created a few setbacks in tuning that could have been solved with revision. The main issue was that the color detecting portion would identify sections of track that were further on the ground, but disconnected from the current line. Much of the tuning time was spent addressing this behavior. However, selecting the desired contour based on proximity to the car as well as color would have freed up more time for tuning and testing other sections.

Finally, the potential field navigator significantly slowed down speed. Using a system to simulate an electromagnetic force caused the car to slow down more often than turn, making objects which were nearly in front of the car significantly harder to navigate. Using an adapted system with minimal pushback and a cosine based turning force would have allowed the car to react more desirably to nearby obstacles.

The node master system also had a few drawbacks. The major disappointment came from the inability of the Robot Operating System to efficiently manage a variable number of nodes, forcing the existing navigation nodes to be edited in the source rather than remain fully modular. Using the remapping system, node input could have been limited with output, preventing nodes from computing when they had no new information. ROS itself, however, does not operate well with significant remapping, making the node master program inherently inefficient.

In other future work, our algorithms can implement these changes, and they should be tested in other environments as well to truly determine adaptability. Additionally, the navigational algorithms could feed into a full motion controller, such as a Pure Pursuit Controller (Snider, 2009). This would allow for smoother motion during driving and more efficient computing, as the navigation and motion planning could be done separately. Additionally, more work should be put into parallelizing the process, allowing faster reaction and more navigational nodes to operate at the same time.

The optimizations made here suggest that the algorithms used cannot reach the speeds needed for emergency response vehicles. Additionally, more quantitative analysis would allow for specific information about the efficiency of individual algorithms as well as the integrated state controller. In its current state, the algorithms do not provide enough data to make significant conclusions about the project.

Acknowledgements

I would like to thank The South Carolina Governor's School for Science and Mathematics, The Summer Program for Research Interns, and Dr. Josh Witten for supporting this research, as well as Dr. Al DeGennaro, Dr. Roy Flannagan for advising me on my report and presentation. Additionally, I would like to express my gratitude for MIT Lincoln Labs Beaver Works for hosting the Beaver Works Summer Institute and hosting me, along with Dr. Robert Shin, Dr. Sertac Karaman, Mr. Kenneth Gregson, Mr. Andrew Fishberg, and Mr. Trevor Henderson for mentoring me during my research.

References

Amidi O. 1990. Integrated Mobile Robot Control. Pittsburgh (PA): Carnegie Mellon University. Available from: http://ri.cmu.edu/publications/integrated-mobile-robot-control/.

Snider J M. 2009. Automatic Steering Methods for Autonomous Automobile Path. Pittsburgh (PA): Carnegie Mellon University. Available from: http://ri.cmu.edu/pub_files/2009/2/Automatic_Steering_Methods_for_Autonomous_Automobile_Path_Tracking.pdf.

Karaman S, Anders A. 2017. RACECAR. Boston (MA). Massachusetts Institute of Technology Lincoln Laboratories. [cited 23 June 2017]. Available from: https://mit-racecar.github.io/.

Gerkey B, Smart W, Quigley M. 2015. Programming Robots With ROS. O'Reilly Media, Sebastopol, CA, United States of America.

Itseez. 2015. Open Source Computer Vision Library [cited 7 June 2017]. Available from: https://github.com/itseez/opencv.

van der Walt S, Colbert C, Varoquaux G. 2011. The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering:22-30.

Anders A. 2017. Visual Servoing. Boston (MA). Massachusetts Institute of Technology Lincoln Laboratories. [cited 18 July 2017]. Available from: https://drive.google.com/open?id=1hqmbU7p8awb4wxLK98CbHQThWyVF372hONwgqUc3UFE.