Low-Cost Localization for Educational Robotic Platforms via an External Fixed-Position Camera

Drew Housten and **William C. Regli**
Department of Computer Science
Drexel University
Philadelphia, PA 19104, USA
{dth29, regli}@drexel.edu

Abstract

One of the major limitations with many off-the-shelf, inexpensive robotics platforms is the lack of a simple way to localize the robot. This is an even bigger issue for educational purposes because the cost of a complete solution is usually an important concern. Many great educational projects utilizing a robot become infeasible because of this limitation. This paper attempts to bridge that gap by presenting a method for performing localization in an unknown environment using a single, fixed-position, external camera. The camera's position does not need to be configured; instead, the localization algorithm will treat image coordinates as features in a topological map. Furthermore, the work is done with an iRobot Roomba vacuum cleaner along with a web-cam to keep overall costs affordable. The low cost of the solution combined with the lack of a complicated configuration requirement helps make the Roomba more useful and accessible to robotics and artificial intelligence education. Results validate the approach and show a significant improvement over relying solely on odometry.

Introduction

This paper addresses the problem of localization on small, inexpensive robotic platforms while keeping the solution costs low and the setup simplistic. The goal of this work is to make low-cost robotics more accessible for educational uses by adding localization capabilities to an otherwise capable robotics platform. The target is to achieve decimeter level accuracy in position using an iRobot Roomba vacuum cleaner robot in an indoor environment. Other researchers have looked at the Roomba as an educational platform (Dickinson et al. 2007; Tribelhorn and Dodds 2007). Other researchers have also looked at the low-cost localization problem using only odometry (OKane 2006), contact sensors (Erickson, O'Kane, and LaValle 2007; Tribelhorn and Dodds 2007), fuducials (Dickinson et al. 2007), on-board cameras (Mecklenburg 2005), and other methods (Barry 2004). This research approaches the problem in a new way by merging two concepts. The first concept is determining the relationship between image coordinates from an external fixed camera and a robot's world coordinates (Rawlinson, Chakravarty, and Jarvis 2004). The second concept is using topological maps for localization (Blanco et al. 2006; Kouzoubov and Austin 2004).

Figure 1: The iRobot Roomba platform with the Sparkfun RooTooth attached

The problem of having a robot know where it is in the environment is not isolated to small robotic platforms. Some researchers have tackled the localization challenge by using Global Positioning Systems (GPS), LIDAR Range Finders, arrays of sonar or infrared sensors (Elfes 1987), odometry through wheel encoders (Borenstein and Feng 1996; OKane 2006), contact sensors (Erickson, O'Kane, and LaValle 2007), cameras (Posner, Schroeter, and Newman 2007; Stronger and Stone 2007), and geometric beacons (Leonard and Durrant-Whyte 1991). Each of these approaches can be applied with and without a known map of the environment, but in either case, they have limitations. A GPS requires a fix to a satellite, which is a difficulty in indoor environments. LIDAR Range Finders are expensive, heavy, and require significant computational power to process. Sonar and Infrared Sensors are relatively short-range and prone to noisy readings that need to be filtered. Odometry tends to drift over time due to slipping and skidding, something that cannot be entirely prevented. Contact sensors require direct interaction with the environment which may affect the world. Cameras require a significant amount of computational power and have limited usefulness. Geometric beacons require augmenting the environment.

iRobot's Roomba is a small, low-cost robotic platform that has the potential for many educational uses (Dickinson et al. 2007). Out of the box, the Roomba has two Drive Motors in a Differential-Drive platform, Vacuum Motors, Brush Motor, Speaker, Wheel Encoders, Bump Sensors, IR Wall Sensors, IR Receiver, Cliff Sensors, Dirt-Detection Sensor, LEDs, Buttons, and several other Battery and Current sensors, all for under 150 US Dollars. Add a bluetooth dongle (Sparkfun RooTooth) for 100 US Dollars, and all the actuators and sensors become accessible via a bluetooth-equipped computer. As is, it is a very capable and inexpensive robotic-platform for education (Tribelhorn and Dodds 2007). However, one lacking feature is the ability to accurately know where it is. Many educational projects rely on the ability to first know where the robot is located in the environment. Even a relatively simple challenge of ”making the robot go to the doorway and beep” requires knowledge about where the robot is, where the doorway is, and when the robot has reached the doorway. Or a simpler task of ”moving forward 1 meter” is just as difficult without a way to localize the robot. A solution to the problem is necessary to make robotics more useful for teaching engineering and artificial intelligence concepts to younger audiences.

Approach

The research looks at a low-cost and simple approach to localization in a 2D indoor environment. A fixed-position camera is used that is external to the robot. The camera is attached to a computer that has access to the Roomba via Bluetooth. Through the Bluetooth connection, the computer is able to direct the Roomba and get the sensor readings back. The camera is directed at the area that the Roomba will travel, but the camera’s specific location is arbitrary. A Topological map is built by fusing together the odometry readings and the image data.

A Topological map is a graph where the vertices are interesting features in the environment and the edges are the transitions between the features. In the research, the coordinates in the image frame are treated as features and the action taken between those coordinates as directed edges. An example of the difference between an Odometry map and a Topological map is provided in Fig 2. In the example, the two maps are equivalent until step 4. The shaded Vertices all represent the same actual position. Due to odometry drift, the Odometry-based map contains two distinct Vertices. On the other hand, the Topological map is able to determine that the Roomba has returned to an already discovered point of interest. Because the Topological map was able to close the loop, the final step is more accurate to the actual position.

The Topological mapping algorithm works by (a) recording the Roomba’s start position in the image frame, (b) making an action, (c) recording the distance and heading traveled, (d) recording the Roomba’s end position in the image frame, and (e) adding an edge to a Topological map where the features are the image frame positions. The localization algorithm involves (a) searching through the Topological map to find a path between the initial position and the current position and (b) calculating the current position by simulating the actions to travel that path. The approach is explained in more detail in this section.

One benefit of this approach is that an a priori environment map is not necessary, because a map is built online as the Roomba is exploring the environment. Another benefit is that the camera’s precise position, angle, and focal length do not need to be calibrated. While this method does require slight augmentation of the environment (adding a camera somewhere in the environment), it is very simple to setup, it is very inexpensive, and it does not require any modification to the Roomba platform itself.

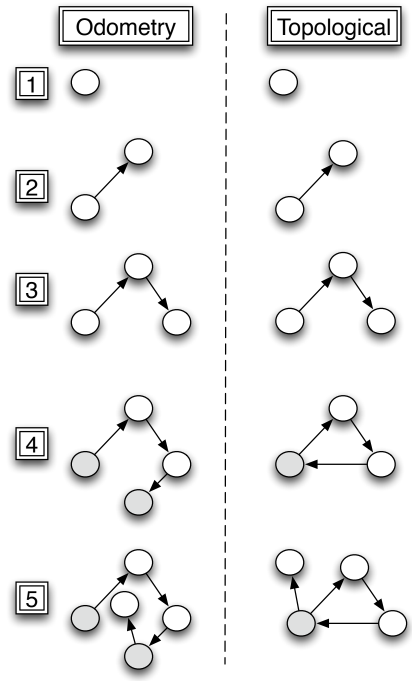

Figure 2: An example showing the difference between an Odometry based map and a Topological Map. In the Odometry based map, the Edges represent the actions taken by the Roomba. In the Topological map, the Edges represent the transition between two points of interest.

Assumptions

For simplicity, this work assumes that all motions consist of a rotation around the COR (center of rotation) followed by a forward translation. Doing so simplifies the models and the transition graph. Since the robot is a differential drive platform, this restriction does not limit where the robot can travel. The work is also based on the assumption that over short distances, the Roomba’s odometry through reading the wheel encoders is fairly accurate (Tribelhorn and Dodds 2007). While not always true, in practice this seems to be a reasonable assumption. The work assumes that a camera can be placed such that the camera’s field of view covers the environment that they robot will be traveling in. For this paper, it is also assumed that the robot will be visible by the camera after completing each action. This limitation can easily be addressed in future work. Finally, the Roomba’s initial position must be known. The position determined by the localization algorithm is relative to this initial pose. An alternative to this limitation is discussed as Future Work.

Odometry Mapping

The Roomba's distance and angle since the sensors were last queried can be accessed through the Roomba Serial Command Interface (SCI) (iRobot 2006). The distance is computed by averaging the encoder values on each wheel and converting the value to millimeters. The angle is computed by taking the difference of the encoder values of each wheel, dividing by 2, and converting the value to radians. Since the actions are limited to rotations followed by forward translations, the Roomba simply needs to be commanded to rotate until the desired angle is reached and then commanded to move forward until the desired distance is reached. This approach does not address the problem of drift, where the Roomba can veer to one way or another while translating, but that limitation can easily be addressed (Discussed as Future Work). Because the Roomba is driven until the encoder values match the commanded actions, the commanded actions and the odometry are equivalent. The Roomba's heading is an accumulation of the angles the Roomba is commanded to travel.

Vision Tracking

The purpose of the camera is to track the position of the Roomba in the image frame. To do so, the tracking algorithm simply looks for some color blob present on the Roomba but not present in the environment. For example, in the test environment, a piece of blue painter's tape was used. The tracking algorithm is first calibrated to look for that color by the user. The user will select one or more regions in a sample image by clicking on the Roomba shown in a GUI. During operation, the tracking algorithm scans through each image pulled from the camera and identifies the regions with an average color close to one of the regions that the user selected. The coordinates of all matching regions are averaged to a single point identifying where the Roomba is in the image frame. While there are other more accurate and quicker algorithms for color tracking, this approach was simplistic.

Topological Mapping

A majority of the work lies in treating the image positions as points of interest in a Topological map. The biggest challenge traditionally with building an on-line topological map is the 'closing the loop' problem (Beevers and Huang 2005). In other words, it is the problem of trying to determine if two features encountered are the same, which means that the robot has traveled a cycle, or if they are actually distinct features. This specific approach gets around that problem because it essentially has an external observer (the camera) telling the robot if it has visited a unique location or not. There are two parts to the topological mapping algorithm: building the map as the robot explores the environment and using the map to determine where the robot is.

Building the Topological Map The robot will make an action. This is limited to a rotation in place followed by a forward translation. The action, along with the image coordinates of the Roomba prior to the move and the image coordinates of the Roomba after the move will be used to create an edge in the Topological Map. The directed edge is created with the end vertex being the image coordinates, the edge distance being the magnitude of the robot's translation, and the heading being the heading the robot was facing while traveling. After the edge is added, a process called *Edge Unification* (described in the following paragraph) will be run to reduce equivalent map edges. Once Edge Unification is complete, the entire process is repeated for a new action and will continue as long as there are more actions to execute. This algorithm is shown in Algorithm 1.

Algorithm 1 Add Action to Topological map

Require: M is a Topological map, A is the action that was last executed, and $prevImagePos$ is the previous image coordinate
- $currImageP \Leftarrow getImageCoordinate()$
- $S \Leftarrow getVertex(M, prevImagePos)$
- $T \Leftarrow getVertex(M, currImagePos)$
- **if** $S =$ NULL **then**
  - $S \Leftarrow createVertex(prevImagePos)$
- **end if**
- **if** $T =$ NULL **then**
  - $T \Leftarrow createVertex(currImagePos)$
- **end if**
- $E \Leftarrow getEdge(M, S, T)$
- **if** $E =$ NULL **then**
  - $E \Leftarrow createEdge(S, T, A)$
  - $addEdge(M, E)$
- **else**
  - $n \Leftarrow getNumReadings(E)$
  - $d \Leftarrow (getDistance(E) * n + getDistance(A))$
  - $d \Leftarrow d/(n + 1)$
  - $setDistance(E, d)$
  - $setNumReadings(E, n + 1)$
- **end if**
- **return** M

Edge Unification is a method of collapsing two edges between the same set of vertices. Two vertices are considered the same if the coordinates are separated by a tolerance. This tolerance is used to prevent a requirement of 'pixel-perfect' tracking. Since all actions are straight line translations, there is only one correct transition between any two adjacent vertices. Therefore, the distances of any two edges between the same set of vertices are averaged. The heading is a little more complicated. The heading could be averaged like the distances, but the heading is a function of all the previous headings on the route to that point. So, the heading will worsen over time as the odometry drifts. This problem could be addressed by tracking heading via the vision, but at the moment it has not been resolved. So, as a result, all headings are time-stamped when the edge is created and the heading of the unified edge is the *older* heading of the two edges.

Localizing Using the Map Determining where a robot is located, given the generated Topological Map, is based on the assumption that over short distances the odometry is

Algorithm 2 Localizing Using the Topological Map

Require: M is a Topological map and $sourceImageP$ is the initial image coordinate
$currImageP \Leftarrow getImageCoordinate()$
$S \Leftarrow getVertex(M, sourceImageP)$
$T \Leftarrow getVertex(M, currImageP)$
$P \Leftarrow getShortestPath(M, S, T)$
$x = 0$
$y = 0$
for all E where E is an edge in P **do**
  $d = getDistance(E)$
  $h = getHeading(E)$
  $x = x + d * cos(h)$
  $y = y + d * sin(h)$
end for
return $createPosition(x, y)$

fairly accurate and it worsens as the robot travels further. Using this assumption, the localization algorithm will first find the closest vertex in the map to the current coordinates of the Roomba in the image frame and then find the shortest path between the initial start vertex and this current vertex. The Roomba's position will be calculated by simulating the actions represented by the edges in this shortest path. Simulating the actions starts with the initial position, determines the position after the first action given the action's distance and heading, and repeats the process until the final action in the path has been used. The result is the estimated current location of the Roomba. This is shown in Algorithm 2.

Results

Experimental Setup

All the code is written in Java using the RoombaComm Java API. The RoombaComm API is a Java library that accesses the Roomba Serial Command Interface (SCI) through the on-board 7-pin serial port (Kurt 2006). The RooTooth Bluetooth dongle acts as the communication channel between the computer running the Java code with the WebCam and the Roomba's 7-pin serial port. Doing so requires no physical modification to the Roomba itself. The Roomba used is iRobot's Scheduler Roomba vacuum cleaner. The camera used is an Apple iSight (Fig. 3), but any webcam would work just as well. The code is run on an Apple 1.25 GHz Powerbook G4.

Figure 3: Apple iSight Webcam used to track the Roomba

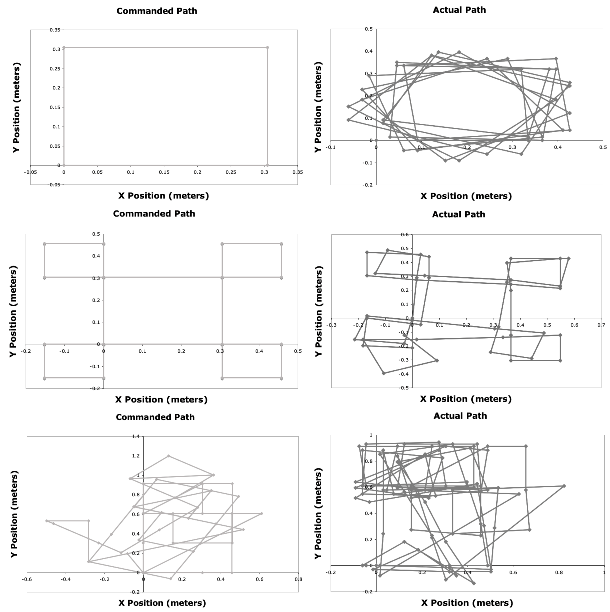

Figure 4: (top-left) Commanded path for Experiment 1 (top-right) Actual path for Experiment 1 (middle-left) Commanded path for Experiment 2 (middle-right) Actual path for Experiment 2 (bottom-left) Commanded path for Experiment 3 (bottom-right) Actual path for Experiment 3

In the tests, the Roomba is given commands to drive around a 1.5 meter by 1.5 meter area which is observed by the fixed iSight camera placed about 1.5 meters high, angled toward the test area. The Roomba is given a series of actions, which consist of a rotation followed by a forward translation. After each action, the actual position of the Roomba, the commanded position of the Roomba, and the position reported by the Topological Camera Localization approach are recorded.

Experiments

Three experiments were run. They are described here and the results are presented. The commanded path and actual path for each experiment are shown in Fig. 4.

1. Square Path: The Roomba makes several circuits of a square path. The Roomba is given a total of 39 actions totaling 12.765 meters of travel.
2. Cloverleaf Path: The Roomba makes two passes around a cloverleaf path, changing directions after the first pass. The Roomba is given a total of 50 actions totaling 10.885 meters of travel.
3. Random Path: The Roomba is given a semi-random path which includes visiting some positions multiple times. The Roomba is given a total of 84 actions totaling 27.489 meters of travel.

Experimental Results

The error between (a) the actual position and the commanded position and (b) the actual position and the topolog-

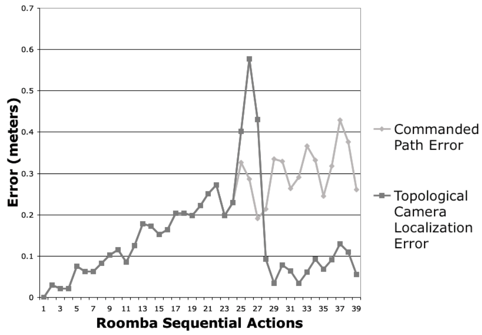

Figure 5: Experiment 1 positional error of commanded position vs. topological camera localization position. The error is computed by taking the straight-line distance between the actual position and the position for each case.

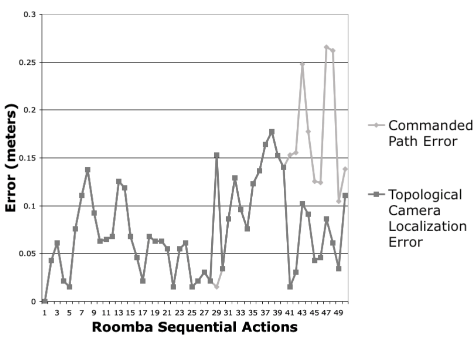

Figure 6: Experiment 2 positional error of commanded position vs. topological camera localization position. The error is computed by taking the straight-line distance between the actual position and the position for each case.

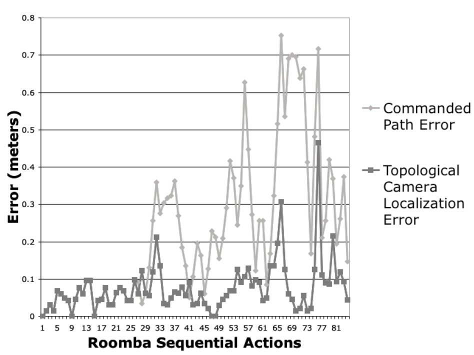

Figure 7: Experiment 3 positional error of commanded position vs. topological camera localization position. The error is computed by taking the straight-line distance between the actual position and the position for each case.

Table 1: Median positional errors for each experiment

Experiment	Commanded Position	Camera Localization Position	Improvement
1: Square	0.204 meters	0.102 meters	50.0%
2: Cloverleaf	0.076 meters	0.064 meters	15.8%
3: Random	0.176 meters	0.062 meters	64.8%

Table 2: Final positional errors for each experiment

Experiment	Commanded Position	Camera Localization Position	Improvement
1: Square	0.261 meters	0.055 meters	78.9%
2: Cloverleaf	0.138 meters	0.111 meters	19.6%
3: Random	0.148 meters	0.044 meters	70.3%

ical camera localization for each experiment are calculated and presented in Fig. 5, Fig. 6, and Fig. 7. The median errors and the path final position errors are shown in Table 1 and Table 2 respectively.

Conclusions and Future Work

At most of the points in the test data, the topological camera localization performed better than relying on odometry only. On average, the camera localization shows significant improvement. Furthermore, if the Roomba continued to move in the same environment, the commanded position error is expected to worsen since the odometry will continue to drift without any method of correction. This trend is already appearing in Fig. 5, Fig. 6, and Fig. 7. On the other hand, if the Roomba travels in the same area, the camera localization is actually expected to *improve* with more actions. This is due to the fact that the topological map will become more densely filled with positions that the Roomba has visited before. However, the camera localization is not perfect. There is still significant error and noisy readings will throw off the position considerably (for example, see the spike in Fig. 5). Also, the method requires the actual path to contain cycles. If there are no cycles, the topological camera localization will perform no better than just using pure odometry. Fortunately this was just the first step to validate the approach. There are many ways to improve the algorithm.

- Vision Tracking: The method to track the robot is simplistic and can be greatly improved both in accuracy as well as speed.
- Filter out bad data: Currently all data is represented in the topological map and no data is filtered out. Doing so should eliminate the occasional poor performance of the localization.
- Tracking Heading: Tracking heading via vision in addition to position will greatly improve the estimated head-

ing between the vertices in the topological map.

- Correct for Odometry drift while translating: The odometry model does not account for angle drift while translating. This can be addressed by monitoring the angle while traveling forward and adjusting the rotational velocity to compensate.
- Use additional paths through the Topological Map: Currently only the shortest path from the start position to the current position of the Roomba is used. If the shortest path was derived from flawed data, any longer paths that might be more correct are ignored. One idea is to find all paths between the start position and the current position, find the final position along each path, and average the final positions weighted by the inverse of the distance. When this method was tried using the sample data, it actually worsened the results, but given a longer test run, this may help. Exploring this and other similar approaches remains an open issue.
- Multiple Cameras: Having multiple cameras in the environment should help by tracking the Roomba independently and then combining the results. While this starts to increase the cost and complexity of the solution, in some situations (for example, large, obscured areas with no single, all-encompassing vantage point) this may be highly beneficial.
- An A Priori Map: The current algorithm requires a known initial position but no other knowledge about the environment. Instead, the algorithm can be adapted to use an A Priori Map of several real-world coordinates mapped to image coordinates. This could be useful if the environment is known beforehand, but the robot's position is not.
- Collapse the Map: After a long runtime, the topological map will become densely populated, which will increase the algorithm's computational time. One solution is to collapse the map after the map reaches a certain density into a series of known points. This essentially creates an A Priori Map (see the above item) and restarts using that map.
- Support more universal motion: The current actions are limited to rotation in place followed by a straight-line translation. The platform is capable of more complete motion. The algorithm could be adapted to support whatever ranges of motion the Roomba can perform.

Finally, the ultimate goal for future work is to continue expanding the capabilities for educational use while keeping the total cost low. The plan is to include this work as part of a toolkit available for pre-college robotics educational programs. The work will continue to evolve as educator feedback is received based on use in active curriculums.

References

Barry, J. 2004. Map Creation and Position Correction for an Off the Shelf Mobile Robotic System. Master's thesis, Virginia Tech, Blacksburg, Virginia.

Beevers, K., and Huang, W. 2005. Loop Closing in Topological Maps. *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on* 4367–4372.

Blanco, J.; Gonzalez, J.; Fernández-Madrigal, J.; and Malaga, S. 2006. Consistent Observation Grouping for Generating Metric-Topological Maps that Improves Robot Localization. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* 818–823.

Borenstein, J., and Feng, L. 1996. Measurement and correction of systematic odometry errors in mobilerobots. *Robotics and Automation, IEEE Transactions on* 12(6):869–880.

Dickinson, B.; Jenkins, O.; Moseley, M.; Bloom, D.; and Hartmann, D. 2007. Roomba pac-man: Teaching autonomous robotics through embodied gaming. In *AAAI Symposium on Robot and Robot Venues: Resources for AI Education*.

Elfes, A. 1987. Sonar-based real-world mapping and navigation. *IEEE. Journal Robotics Automation* 3(3):249–265.

Erickson, L.; O'Kane, J.; and LaValle, S. 2007. Probabilistic localization using only a clock and a contact sensor. *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Submitted, under review.

iRobot. 2006. Roomba Serial Command Interface (SCI) Specification. [Online]. Available: http://www.irobot.com/hacker.

Kouzoubov, K., and Austin, D. 2004. Hybrid topological/metric approach to SLAM. *Robotics and Automation, 2004 IEEE International Conference on* 1:872–877.

Kurt, T. 2006. *Hacking Roomba: ExtremeTech*. John Wiley & Sons, Inc. New York, NY, USA.

Leonard, J., and Durrant-Whyte, H. 1991. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on* 7(3):376–382.

Mecklenburg, P. 2005. Roomba SLAM. [Online]. Available: http://www.paulmecklenburg.org/school/roomba.

OKane, J. 2006. Global localization using odometry. *Robotics and Automation, 2006 IEEE International Conference on* 37–42.

Posner, I.; Schroeter, D.; and Newman, P. 2007. Describing Composite Urban Workspaces. *Robotics and Automation, 2007 IEEE International Conference on* 4962–4968.

Rawlinson, D.; Chakravarty, P.; and Jarvis, R. 2004. Distributed Visual Servoing of a Mobile Robot for Surveillance Applications. *Australasian Conference on Robotics and Automation*.

Stronger, D., and Stone, P. 2007. A Comparison of Two Approaches for Vision and Self-Localization on a Mobile Robot. *Robotics and Automation, 2007 IEEE International Conference on* 3915–3920.

Tribelhorn, B., and Dodds, Z. 2007. Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education. *Robotics and Automation, 2007 IEEE International Conference on* 1393–1399.