Performance Metrics for Robot Coverage Tasks

Sylvia C. Wong, Lee Middleton and Bruce A. MacDonald
Department of Electrical and Electronic Engineering
The University of Auckland
Auckland, New Zealand

{s.wong,l.middleton,b.macdonald}@auckland.ac.nz

Abstract

Applications that require a robot to cover an area can be described as complete coverage navigation tasks. Examples of this type of navigation are vacuuming, surface coating, and systematic foraging. Performance measures are often neglected in this area of research. This paper presents two metrics for measuring performance of robot coverage tasks and applied them to a real robot coverage experiment. The first metric measures percentage of coverage using computer vision techniques. The second metric uses distance travelled by the robot. It is found that percentage of coverage is a good performance indicator if physical limitations of the robot is taken into account; while distance travelled by itself is a poor indicator because it completely ignores the amount of area covered.

1 Introduction

In tasks such as vacuum cleaning, painting and landmine searching, a mobile robot must visit all reachable floor surface in an enclosed region. This type of navigation can be described as complete coverage navigation. Here, a robot carries out a coverage task that ensures no area is missed.

It is important to have a measure on how well the algorithm performs. Generally, the mesurement of performance is well studied in the area of localisation [Duckett and Nehmzow, 1998, Nehmzow, 2001]. However, this is often neglected in discussion of research on robot coverage navigation. For example, [Acar *et al.*, 2001] and [Butler *et al.*, 1999] gave only a theoretical analysis of their algorithms. Others opted to include diagrams showing how a given environment might be covered by their algorithms [González *et al.*, 1996, Luo and Yang, 2001, Zelinsky *et al.*, 1993].

The important performance measures of this type of algorithm are the effectiveness and the efficiency of coverage. In [Gabriely and Rimon, 2002], the simulated test environment was divided into equal sized cells. The effectiveness of coverage was then measured as the ratio of the number of cells that were visited at least once over the total number of unoccupied cells. For experiments on real robots, [Ulrich *et al.*, 1997] covered the test area with sawdust and estimated the amount of sawdust left afterwards. This requires the test platform to be equipped with a dustbuster. It is also error prone as estimation of the amount of dust is not an easy task.

For efficiency of coverage in simulation, [Gabriely and Rimon, 2002] defined any cells that were repeatedly covered as undesirable and thus a good solution would minimise the amount of repeated coverage. This concept of repeated visits was also mentioned in [Zelinsky *et al.*, 1993]. No reference can be found for efficiency measures for experiments on real robots.

This paper presents two methods for measuring the performance of complete coverage algorithms that can be used for the research we are doing in this area. Section 3 describes a method to measure percentage of coverage (effectiveness) using computer vision techniques. Section 4 discusses how efficiency is measured using dead reckoning information. These metrics were tested on the topological coverage algorithm described in section 5. Results and discussion can be found in sections 6 and 7.

2 Experimental Setup

The performace metrics in this paper is tested on a miniature Khepera robot [Mondada *et al.*, 1993] controlled by a Linux PC via the serial port. The robot is 53mm in diameter. It is equipped with 8 infra-red proximity sensors which can detect objects up to 30 to 40mm away. It is also equipped with optical wheel encoders for dead reckoning. The testing is done in a wooden tray of 75x75cm, which is appropriate for the size of the robot. Ten wooden blocks of 10x10cm sides and a separator are available to act as objects within the environment. Figure 1 shows a picture of the tray with the obstacles and the robot inside.

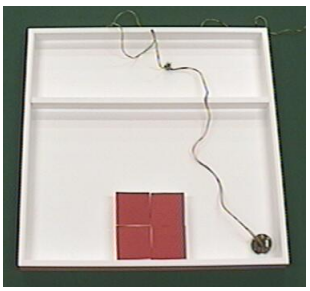


Figure 1: The miniature Khepera robot in a 75x75cm wooden tray.

3 Effectiveness: Coverage Percentage

Computer vision was used to measure the effectiveness of coverage by a robot. A wall mounted camera was used to capture a movie of the robot's progress within the environment. The approach involves superimposing all the captured frames to compute the covered area. The equipment is configured as shown in figure 2 and in our case exploited a setup which is also being employed for *robosoccer* within the department. Images from the camera were captured on a computer running Linux and the `Xawtv` software. (This is a different PC from the one controlling the robot as described in section 2). Figure 1 shows a sample image taken from this wall mounted camera. It can be seen that the image contains perspective distortion. This results in the original rectilinear area becoming a trapezium with the regions nearer the camera (at the bottom of the figure) being larger than the regions further away.

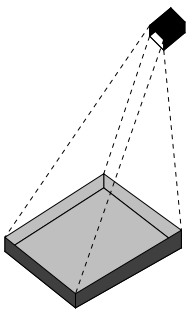


Figure 2: Mounted camera capturing frames of robot's movement.

The area estimation system is detailed in figure 3. There are four main processes. The first is capturing the navigation process using the camera and extracting individual frames. The second is to condense these frames into a coverage map which illustrates the total region that the robot covered. The third is removing the perspective distortion. The final process is calculating the effectiveness metric. With the exception of the first step, which can trivially be performed

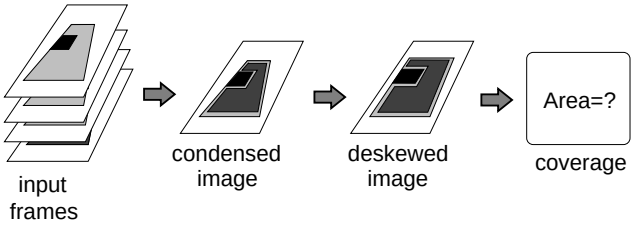


Figure 3: System to estimate the coverage area.

by `Xawtv`, the implementation details of the remaining steps will now be discussed.

3.1 Producing the coverage map

Moving on an average of 0.012 m/s, the robot takes approximately 5 minutes to cover an average environment inside the tray. Thus even at low frame rates, several thousand frames are generated. For analysis of the coverage information it is essential to reduce this information to a single image which contains a superposition of all the places the robot has visited. This process requires two sorts of information – the individual frames that were captured $f_i(x, y)$, and a reference image $f_r(x, y)$. The reference image is identical to that in figure 1 except that the robot has been removed. Before processing both f_i and f_r were reduced from colour images to greyscale ones. This process involved a weighted sum of the individual red, green and blue components of the colour [CCIR Recommendation 601-2, 1985]:

$$I = 0.2989r + 0.5870g + 0.1140b \tag{1}$$

This returns the greyscale intensity I for a colour (r, g, b). The algorithm to generate the composite map consists of two distinct steps. Firstly, a series of difference images are generated and secondly these are combined to produce the coverage map. The difference images are found by subtracting each frame from the reference image. For all N frames the difference images, g_i, can be computed as follows :

$$(x_j, y_j) = h(f_i(x_j, y_j) - f_r(x_j, y_j)) \qquad \forall i \in [0, N-1] \tag{2}$$

Here h is a threshold function and (x_j, y_j) are coordinates in the image. All the operations are performed pointwise on the associated images. The threshold function is a step response which is defined as follows :

$$h(v) = \begin{cases} 1 & , v > 127 \\ 0 & , else \end{cases} \tag{3}$$

Given all the difference images the coverage map can be produced by performing a pointwise `OR` of all the images. This methodology works as the main point

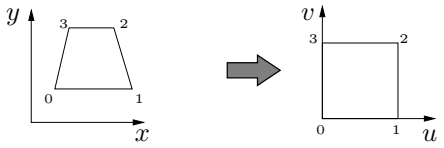


Figure 4: Quadrilateral to square mapping.

of difference between successive frames is the robot. As the thresholded images, g_i, are effectively a binary representation of this difference then a logical `OR` will yield a resulting image which is a superposition of the robots locations.

3.2 Correcting the perspective warp

The images captured by the camera shows a effect known as a perspective warping (figure 1). This means that distant lines are shortened when compared to closer lines. This shortening effect makes it difficult to estimate the coverage area and so it is essential to correct for this effect. This can be performed using an inverse perspective transform to map the quadrilateral into a unit region as shown in figure 4. This mapping makes the assumption that the original object, in this case the tray, is square in shape. The perspective transform described in this section follows the work of [Wolberg, 1990].

To make the mathematics simpler this discussion will originally examine the reverse of the transform shown in figure 4. The forward perspective can be written:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{4}$$

Here x, y, u and v are coordinates as shown in figure 4. w is the perspective. It can be simply written as $\mathbf{x}' = \mathbf{A}\mathbf{u}$. Euclidean coordinates can be computed using equation (4) and performing the following substitutions :

$$x = \frac{x'}{w'} = \frac{a_{11}u + a_{12}v + a_{13}w}{a_{31}u + a_{32}v + a_{33}w} \tag{5}$$

$$y = \frac{y'}{w'} = \frac{a_{21}u + a_{22}v + a_{23}w}{a_{31}u + a_{32}v + a_{33}w} \tag{6}$$

For most applications w is generally 1.

For arbitrary systems (quadrilateral to quadrilateral) solving this equation will have 8 degrees of freedom (two for each point in the quadrilateral). However, for the situation here the uv plane is a unit square. Thus :

$$\begin{aligned} (0,0) &\rightarrow (x_0, y_0) \\ (1,0) &\rightarrow (x_1, y_1) \\ (1,1) &\rightarrow (x_2, y_2) \\ (0,1) &\rightarrow (x_3, y_3) \end{aligned}$$

This gives the following values for the elements of the transformation matrix :

$$\mathbf{A} = \begin{bmatrix} x_1 - x_0 + a_{31}x_1 & x_3 - x_0 + a_{32}x_3 & x_0 \\ y_1 - y_0 + a_{31}y_1 & y_3 - y_0 + a_{32}y_3 & y_0 \\ \frac{\Delta x_3 \Delta y_2 - \Delta y_3 \Delta x_2}{\Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2} & \frac{\Delta x_1 \Delta y_3 - \Delta y_1 \Delta x_3}{\Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2} & 1 \end{bmatrix} \tag{7}$$

To compute the reverse of this transform requires $\mathbf{A}^{-1}$ be found. This can be found by observing that $\mathbf{A}^{-1} = \frac{\text{adj}\,\mathbf{A}}{\det \mathbf{A}}$. Now as $\det \mathbf{A}$ is a scalar quantity then $\text{adj}\,\mathbf{A}$ can be used as an approximation of $\mathbf{A}^{-1}$ so long as suitable scaling is performed on the coordinates.

To employ this transformation on the image requires that the corner points of the quadrilateral be known. This could be computed via image processing and the methodology of edge detection. However, as they are fixed in all frames they were found by empirical examination. Once found, these points were used to generate a grid of points at which to sample the image. Neighbouring points yield a quadrilateral in the xy plane and once deskewed they form a rectangle in the uv plane. For each quadrilateral the intensity was computed using bi-linear interpolation (average of the intensities of the nearest pixels). So long as the density of points in the grid is sufficiently large then this method will yield an accurate image with the perspective warp removed.

3.3 Computing the coverage

Computation of the coverage information is a relatively simple process. It requires that three things are known – the total area of the tray (A_t), the total area of the obstacles (A_o), and the total area of the robots path (A_r). From these three measures the percentage coverage performed by the robot can be computed by :

$$C = \frac{100 A_r}{A_t - A_o} \tag{8}$$

The area of the tray can be found from the deskewed image as the number of pixels in the image. For example for an image that has width, w, and height h then $A_t = wh$. The obstacles areas can be found from the reference image. Notice, in figure 1 that the obstacle is covered with paper. This paper is red to aid automated identification. By performing the the deskewing process on the reference image and filtering out all pixels which are not red in colour the area of the obstacles can be simply found. The coverage map, as described in section 3.1, consists of binary values which denote whether the robot has travelled to a position in the image or not. Hence, A_r can be found by summing the number of values in the coverage map. Once found these three values can be substituted in equation (8) to compute the covered area.

4 Efficiency: Distance Travelled

The algorithm described in section 5 moves the robot in a zigzag pattern. If there is no overlap between neighbouring strips, the area to be covered can be divided into horizontal strips. The width of an individual strip is the width of the robot. The minimum path length to cover a given region is thus the sum of the lengths of these strips. Figure 5 illustrates this idea.

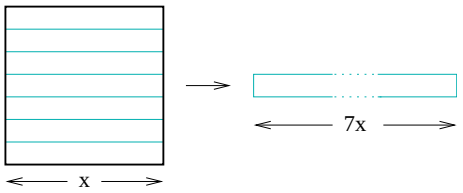


Figure 5: Idealised minimal path length.

This minimum path length will be shorter than or equal to the realisable optimal path length needed to cover a given environment. In the case of a rectangular environment with no obstacles, the minimum path length calculated this way is the same as the realisable optimal path length. More complex environments with polygonal objects will have longer realisable paths. This is because of the distance required to travel between different regions will invariably involve covering already visited floor area. In simpler terms, calculating the optimal path is equivalent to solving the Travelling Salesman Problem where each floor position is a city to visit.

The efficiency of the coverage (E) can be calculated by :

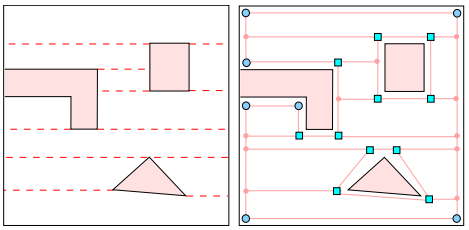
$$E = \frac{D_a}{D_m \times C} \tag{9}$$

Here, D_m is the idealised minimum path length and D_a is the actual distance travelled by the robot calculated from dead reckoning information. C is the coverage from equation (8). $D_m \times C$ gives the minimum path length scaled by the amount of coverage. For example, a robot achieved 50% coverage of a given environment. The minimum path length of the area covered is thus $D_m \times 0.5\%$. If D_m is not scaled by C, a robot that poorly covers a given environment but travels a little will have a low $\frac{D_a}{D_m}$ ratio.

This concept of idealised minimum path length is analogous to the use of repeatedly covered cells in simulation by [Gabriely and Rimon, 2002]. There, the number of repeatedly covered cells is E and the minimum path is the path that covers all unoccupied cells once and only once.

5 Case Study: Topological Coverage Algorithm

The effectiveness and efficiency metrics was tested using the topological coverage algorithm in [Wong *et al.*, 2000]. The algorithm works by subdiving any rectangular region with polygonal objects within using the objects as division points (figure 6(a)). This subdivision can be represented with a



(a) (b)

Figure 6: (a) Segmentation into subregions. (b) Subregion representation using planar graphs.

planar graph, with the nodes representing landmarks and edges representing travel edges (figure 6(b)). With a graph, subregions are defined by edges.

Starting at a corner, the robot covers the first subregion with a zigzag pattern (figure 7). When a landmark is reached, the subregion is fully covered as objects mark the boundaries between subregions (figure 6(b)). The internal representation is updated with this newly discovered node. The robot finishes covering the strip in the zigzag pattern that it is currently covering. This is to expose all landmarks on the same boundary. The constructed graph is then searched to find any uncovered subregion. This process is repeated until all subregions are covered.

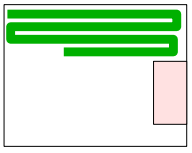


Figure 7: Coverage with a zigzag pattern.

6 Results

The coverage algorithm was implemented on the Khepera robot and used to cover the tray described in section 2. Figure 8 shows the dimensions of the environment.

The coverage map was computed and the image had the perspective warping removed using the method discussed in section 3. The resulting image is shown in

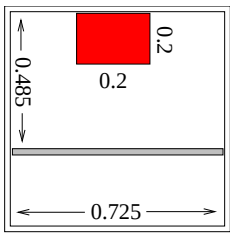


Figure 8: Sample environment. All measurements are in metres.

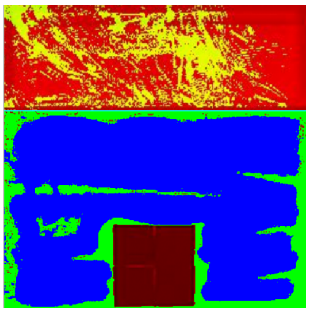


Figure 9: The deskewed and false coloured coverage map.

figure 9. False colour is used to identify the various regions of importance. In the figure, the blue area is the area covered by the robot and the maroon region is the obstacle. The final image is of size 512×512 but the reachable area is of size 512×334. This yields the following areas :

$$A_t = 171008 \text{ pixels}$$
$$A_o = 20000 \text{ pixels}$$
$$A_r = 118586 \text{ pixels}$$

From these figures and equation (8), the total coverage for the robot is 79%.

Using the dimensions shown in figure 8, the total floor area is found to be 0.3116 m^2. The idealised minimum path length is thus :

$$D_m = \frac{0.3116\text{m}^2}{0.053\text{m}} = 5.88\text{m}$$

Here 0.053 m is the diameter of the robot. D_a is estimated to be 6.00 m^2 from dead reckoning calculations. Thus using equation (9) :

$$E = \frac{6.00}{5.88 \times 0.79} = 1.29$$

7 Discussion

It can be seen from figure 9 that most of the floor areas missed by the robot are around the border of the tray and obstacles. This is due to physical limitations of driving the robot next to objects. If this limitation is taken into account, this reachable floor area will always be smaller then A_t. This means that the coverage algorithm performance is better than the 79% implies. To obtain a more realistic coverage figure, the dimensions of the tray can be shrunk and the obstacles within can be grown to accommodate for this physical limitation.

In the calculation of E, D_m is scaled with the percentage coverage C. This means E is not affected by the inaccessible border region. Acutal distance travelled D_a is compared with minimum path length of the region covered ($D_m \times C$). The concept of minimum path is used instead of the optimal coverage path because finding the optimal path is akin to the travelling salesman problem and thus NP-hard. As the minimum path should always be equal or shorter than the optimal path, E will be a conservative estimation of the algorithm's efficiency.

The performance metrics are independent of the algorithm used to navigate the robot. The effectiveness measure uses a movie that capture the coverage process from a fix position. The efficiency measure uses odometry information and the effectiveness. This means these metrics can be used to compare the performance of different algorithms easily.

8 Conclusions

This paper presented two performance metrics for measuring the effectiveness and efficiency of any given coverage algorithm. Effectiveness is measured using computer vision techniques on a movie that captures the robot's movements. Efficiency was calculated using the distance travelled obtained from odometry information. It is found that both metrics are suitable for measuring the performance of robot coverage tasks.

References

[Acar *et al.*, 2001] E. U. Acar, H. Choset, and P. N. Atkar. Complete sensor-based coverage with extended-range detectors: a hierarchical decomposition in terms of critical points and voronoi diagrams. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1305–1311, 2001.

[Butler *et al.*, 1999] Zack J. Butler, Alfred A. Rizzi, and Ralph L. Hollis. Contact sensor-based coverage of rectilinear environments. In *Proceedings IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, pages 266–271, 1999.

[Duckett and Nehmzow, 1998] Tom Duckett and Ulrich Nehmzow. Mobile robot self-localisation and

measurement of performance in middle-scale environments. *Robotics and Autonomous Systems*, 24:57–69, 1998.

[Gabriely and Rimon, 2002] Yoav Gabriely and Elon Rimon. Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 954–960, Washington, DC, May 2002.

[González *et al.*, 1996] E. González, A. Suárez, C. Moreno, and F. Artigue. Complementary regions: a surface filling algorithm. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 909–914, Minneapolis, Minnesota, April 1996.

[Luo and Yang, 2001] Chaomin Luo and Simon X. Yang. Complete coverage path planning with designated starting and target locations using a neural network paradigm. In *Proceedings 8th International Conference on Neural Information Processing ICONIP*, pages 1157–1163, Shanghai, China, November 2001.

[Mondada *et al.*, 1993] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturisation: A tool for investigation in control algorithms. In T. Yoshikawa and F. Miyazaki, editors, *Proceedings of the Third International Symposium on Experimental Robotics*, pages 501–513, 1993.

[Nehmzow, 2001] Ulrich Nehmzow. Quantitative analysis of robot-environment interaction - on the difference between simulations and the real thing. In *Proceedings Eurobot*, 2001.

[Ulrich *et al.*, 1997] Iwan R. Ulrich, Francesco Mondada, and J. D. Nicoud. Autonomous vacuum cleaner. *Robotics and Autonomous Systems*, 19(3–4):233–245, March 1997.

[CCIR Recommendation 601-2, 1985] CCIR Recommendation 601-2. *Encoding parameters of digital television for studios.* Internation Radio Consultive Committee (ITU), December 1985. Document 11/1041-E.

[Wolberg, 1990] George Wolberg. *Digital Image Warping.* IEEE Computer Society Press, 1990.

[Wong *et al.*, 2000] Sylvia Wong, George Coghill, and Bruce MacDonald. Landmark-based world model for autonomous vacuuming robots. In *Proceedings International ICSC Congress on Intelligent Systems and Applications (ISA)*, Wollongong, Australia, 2000.

[Zelinsky *et al.*, 1993] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robots. In *International Conference on Advanced Robotics ICAR*, Tokyo, Japan, November 1993.