

# Hybrid grids and the Homing Robot

Joseph Rabinoff

*Department of Mathematics, Harvard University, PO Box 322, Fairfield, IA 52556*

---

## Abstract

In their paper [2], Wongngamnit and Angluin introduced a memory-efficient robot, called the Homing Robot, which localizes in an occupancy grid. We present a more general class of grids called hybrid grids, and establish the least upper bound for the number of moves the robot takes to localize. We also state analogous results for a hexagonal tiling.

*Key words:* Algorithms, Hybrid grid, Robot localization

---

## 1 Introduction

Suppose we place a robot in a rectangular grid of square cells, with some cells blocked. The robot can only see the eight cells in its immediate neighborhood, and can only move north, south, east, and west, one square at a time. The robot knows the maximum dimensions of the grid, but it does not know where it has been placed. We assign it the task of determining its coordinates while placing constraints on the memory it can use. The question is, how many moves must the robot take to *localize*, that is, to determine its position?

In [2], Wongngamnit and Angluin introduced an algorithm accomplishing this task, which they named the Homing Robot. They proved that the Homing Robot localizes in such a grid in under  $4.5n$  moves (where  $n$  is the number of free cells), using only  $O(\log n)$  bits of memory. In a subsequent paper [3] they reduced the bound to  $4n$ , but they did not show that the bound is sharp; indeed, they found a class of grids in which the robot localizes in  $3n - o(n)$  moves. In this paper, we show that  $3n$  is the actual least upper bound for the Homing Robot.

---

*Email address:* `rabinoff@post.harvard.edu` (Joseph Rabinoff).

We first introduce hybrid grids, and prove some results about them. We then generalize the Homing Robot to such grids, and prove that  $3n$  is indeed the least upper bound. (This generalization is not frivolous — the concept of a hybrid grid is a necessary element in this proof.) Finally, we demonstrate the flexibility of our approach by presenting the easy analogs of our proofs in a hexagonal tiling.

## 2 Hybrid grids

### 2.1 Definitions

We first define a *hybrid grid* to be a generalization of an occupancy grid. An occupancy grid is a (finite) rectangular arrangement of adjacent square *cells*, with some of these cells *blocked* and some *free*. In a hybrid grid we also allow the edges that border the cells to be blocked (see Figure 1). We do not allow an edge that borders a blocked cell to be called blocked as well, as such a designation would be superfluous. We call an arbitrary set of blocked cells and lines a *barricade*. We define the *neighborhood* of a free cell to be the eight adjacent cells and the twelve incident lines as shown in Figure 2. As in [2], we say that two cells are (*edge-*)*adjacent* if they share an (unblocked) edge, and we define a region of free or blocked cells to be (*edge-*)*connected* if there is an edge-adjacent path between any two cells in the region. We say that a grid is *connected* if the free cells are connected.

We depart from the treatment of obstacles in [2], in order to allow for the added generality of hybrid grids. We define an *obstacle* to be a barricade that a robot can circumnavigate. The definition of circumnavigation is immediate: we allow the robot, which we call the *circumnavigation robot*, to have an orientation, that is, we allow it to face either north, south, east, or west. The robot moves forward (that is, in the direction it is facing) from cell to cell, always keeping the obstacle on the left, until it returns to its starting cell (see Figure 1). We should note that all robots in this paper will be *edge robots*, which means that they are only allowed to move in the four cardinal directions — with this definition, the connected grids are exactly those grids such that there is a path between any two free squares that a robot can traverse.

This definition of an obstacle suggests a distinction. We define the *border* of an obstacle to be all those free cells that have a part of the obstacle in their neighborhood. If the circumnavigation robot circumnavigates the obstacle in a counter-clockwise direction, then we say it is an *interior* obstacle, because the obstacle is inside the border; otherwise, it is an *exterior* obstacle. In a connected occupancy grid, these correspond respectively to the definitions of

*normal* and *boundary* obstacles in [2]. We also call the exterior obstacle in a connected grid the *boundary obstacle* (note that there is only one exterior obstacle if the grid is connected).

## 2.2 Connectedness

The definition of obstacle also suggests an algorithm (called the *connectedness algorithm*) for checking if a grid is connected. For every free cell, we check if the edge or cell to the north, south, east, and west is blocked. If, for instance, the north is blocked, we draw an arrow along the north edge, pointing in the clockwise direction (relative to the center of the free cell, right in this case). See Figure 3. Note that every arrow is thus associated with precisely one free space and one obstacle, with the free space on the right. When this is accomplished for all cells, we start with an arbitrary arrow and follow the arrows until we have formed a loop. This process requires a bit of explanation, as there is ambiguity if there is a choice of arrows to follow that all have the same base point. We deal with this ambiguity as follows: we follow the first outgoing arrow in the counter-clockwise direction from our entering arrow (if there is an outgoing arrow on the same edge, we designate it the *last* arrow, not the first). When we have thus arranged all of the arrows into loops, we check if each loop is oriented clockwise or counter-clockwise, which can be accomplished by noting the direction of a northernmost horizontal arrow.

**Lemma 1** *The number of clockwise loops is the number of connected regions, and the number of counter-clockwise loops is the number of interior obstacles.*

**PROOF.** First we show that a connected region gives a clockwise loop. It is easy to see that such a region gives one of the loops that we found above: if there is an ambiguity about which arrow to follow, then there are two barricades that meet at a point, and we always follow the outside of the obstacle. This loop is clockwise since a northernmost edge has a free cell below it. Similarly, an interior obstacle gives a counterclockwise loop. Since every arrow in a loop always has a free cell on the right, it is obvious that a clockwise loop gives a connected region, and since each arrow has an obstacle on the left, a counterclockwise loop must give an interior obstacle.

Let  $n$  be the number of free cells. The connectedness algorithm requires  $O(n)$  bits of memory for the arrows and their directions, and to arrange them into loops; it sets and checks each bit at most once, so it is also  $O(n)$  complexity in time.

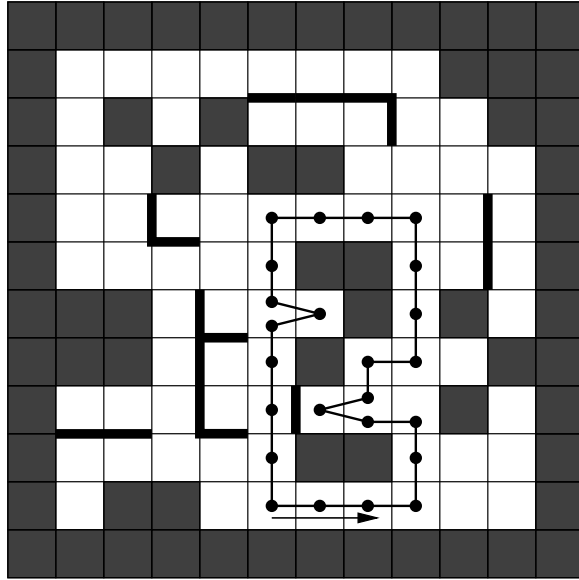


Fig. 1. An example of a connected hybrid grid. The lines around the center obstacle denote the path an edge circumnavigation robot would use to circumnavigate it, and the arrow indicates the direction. There are three interior obstacles and one exterior obstacle.

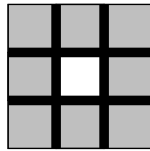


Fig. 2. The shaded cells and the thick lines are the neighborhood of the center cell.

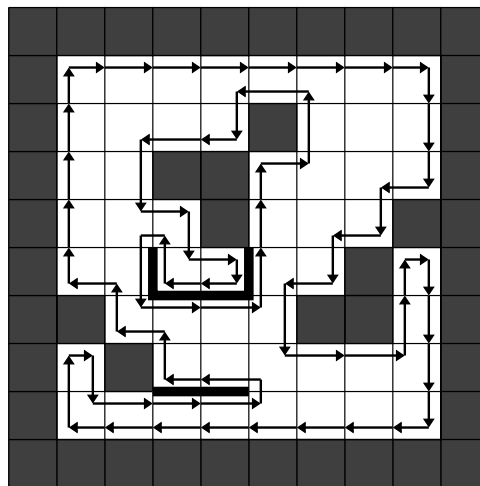


Fig. 3. The arrows one would draw in the connectedness algorithm, offset slightly from the lines on which they should be drawn for clarity. In this figure, the interior obstacle has perimeter 20.

### 3 One obstacle

For the rest of the paper we assume that all grids are connected; we keep the assumption that all robots are edge robots. Let  $\mathcal{O}$  be an (interior or exterior) obstacle, and let  $m = m(\mathcal{O})$  be the number of moves it takes for the circumnavigation robot to circumnavigate  $\mathcal{O}$ .

#### 3.1 Properties

We first prove two interesting lemmas about single obstacles, which will be vital later when we reduce the problem of finding a bound for the entire grid to finding this much easier single-obstacle bound. Drawing arrows as in the connectedness algorithm, we define the *perimeter*  $p = p(\mathcal{O})$  of  $\mathcal{O}$  to be the number of arrows drawn around  $\mathcal{O}$  (see Figure 3).

**Lemma 2** *If  $\mathcal{O}$  is an interior (resp. exterior) obstacle, then  $m = p + 4$  (resp.  $m = p - 4$ ).*

**PROOF.** Let  $l$  be the number of left turns the circumnavigation robot makes while circumnavigating  $\mathcal{O}$ , and let  $r$  be the number of right turns. Since the robot turns if and only if there is a turn in the arrows on the perimeter,  $l$  and  $r$  are also the numbers of left and right turns of the perimeter loop. Now, two straight arrows in the perimeter give two (straight) moves, but a left turn gives a convex corner, which adds one move, and a right turn gives a concave corner, which subtracts one move (see Figure 4). Thus we see that  $m = p + l - r$ . But since the perimeter arrows we drew form a loop (they go around the obstacle), we know that  $r = l \pm 4$ , with the sign depending on whether the loop is clockwise or counter-clockwise: the robot makes four extra *left* turns if the loop is counter-clockwise (so  $\mathcal{O}$  is interior), and four extra right turns otherwise. This gives  $m = p \pm 4$ , with the sign consistent with the statement of the lemma.

It is useful to be able to calculate  $m(\mathcal{O})$ , a global property, as a sum of local quantities. Let  $m(s)$  denote the number of times the circumnavigation robot moves into cell  $s$ . It is useful here to define a barricade to be *obstacle-connected* if a circumnavigation robot would circumnavigate that barricade as an obstacle, were it alone in an empty grid (in other words, the barricade is a single obstacle in itself).

**Lemma 3**  *$m(s)$  is the number of distinct obstacle-connected barricades in the neighborhood of  $s$ .*

**PROOF.** By the number of distinct obstacle-connected barricades in the neighborhood of  $s$ , we mean the number of obstacles in an empty grid containing only  $s$  and its neighborhood (of course two distinct obstacle-connected barricades are part of the same obstacle in the global picture). It is useful to think of  $m(s)$  as the number of trips the robot takes *through*  $s$ . Any one trip will enter through one edge and leave through another (possibly the same). No two trips can enter through, or leave through, the same edge, since they would have to be following the same part of the obstacle on the left. Everything to the left of one trip is one obstacle-connected barricade, and since the trip enters and leaves through two edges, it separates that barricade from the others by a whole cell. See Figure 5 for examples of how  $m(s)$  is determined by the neighborhood of  $s$ .

### 3.2 Main result

This result holds for any hybrid interior or exterior obstacle  $\mathcal{O}$ . Let  $b = b(\mathcal{O})$  be the number of border cells of  $\mathcal{O}$ .

**Theorem 4**  $m < 2b$ .

**PROOF.** As in Lemma 2, define  $l$  and  $r$  to be the number of left and right turns the robot makes during circumnavigation, respectively; note that each turn is associated with exactly one cell. Define  $l(s)$  and  $r(s)$  to be the number of left and right turns the robot makes in cell  $s$ . Let  $m_i$ ,  $1 \leq i \leq 4$ , be the number of cells that the robot visits  $i$  times, as in [2]. The idea of this proof is to use the fact that cells  $s$  with  $m(s) > 2$  require left turns, and that right turns are usually in cells with  $m(s) < 2$ . Since the number of left turns is roughly the number of right turns, these quantities balance out.

**Lemma 5**  $m_1 - m_3 - 2m_4 - 2 \geq 0$ .

**PROOF.** First we bound  $l$  below and  $r$  above by the  $m_i$ ; see Figure 5 for these worst-case scenarios.

- If  $m(s) = 1$  then  $r(s) \leq 2$  and  $l(s) \geq 0$ .
- It is useful to divide the cells  $s$  with  $m(s) = 2$  into two quantities,  $m_2^1$  and  $m_2^2$ , according to whether the cell contains a right turn. Let  $m_2^1$  be the number of cells  $s$  with  $m(s) = 2$  and  $r(s) = 1$  (note that  $r(s) \leq 1$  in such a cell), and write  $m_2 = m_2^1 + m_2^2$ . In the first type of cell, we see that  $l(s) = 1$ , and in the second type,  $l(s) \geq 0$ .
- If  $m(s) = 3$  then  $l(s) \geq 2$  and  $r(s) = 0$ .

- If  $m(s) = 4$  then  $l(s) = 4$  and  $r(s) = 0$ .

We therefore have

$$\begin{aligned} l &\geq m_2^1 + 2m_3 + 4m_4 \\ r &\leq 2m_1 + m_2^1. \end{aligned} \tag{1}$$

Now, if  $\mathcal{O}$  is an interior obstacle, we know that  $l - r = 4$ , i.e. there are four extra left turns (cf. Lemma 2). Consider the northernmost cells that the robot visits while circumnavigating. The easternmost and westernmost of these are both left turns in cells  $s$  with  $m(s) = 1$ . We can find two more such cells at the south of the obstacle. Therefore we can improve our bounds to

$$\begin{aligned} l &\geq m_2^1 + 2m_3 + 4m_4 + 4 \\ r &\leq 2(m_1 - 4) + m_2^1 < 2m_1 + m_2^1 - 4, \end{aligned} \tag{2}$$

so we have

$$\begin{aligned} 4 = l - r &\geq -2m_1 + 2m_3 + 4m_4 + 8 \\ \implies 0 &\leq m_1 - m_3 - 2m_4 - 2. \end{aligned} \tag{3}$$

If  $\mathcal{O}$  is an exterior obstacle, we know  $l - r = -4$ , so we immediately have that

$$\begin{aligned} -4 = l - r &\geq -2m_1 + 2m_3 + 4m_4 \\ \implies 0 &\leq m_1 - m_3 - 2m_4 - 2. \end{aligned} \tag{4}$$

The rest of the proof of Theorem 4 is immediate. We know that  $b = m_1 + m_2 + m_3 + m_4$  and  $m = m_1 + 2m_2 + 3m_3 + 4m_4$ , so, using Lemma 5, we have that

$$\begin{aligned} m &= m_1 + 2m_2 + 3m_3 + 4m_4 \\ &\leq 2m_1 + 2m_2 + 2m_3 + 2m_4 - 2 < 2b. \end{aligned} \tag{5}$$

Theorem 4 is a least upper bound: define  $\mathcal{C}_{w,h}$  to be the comb shape of width  $w$  and height  $h$ , as shown in Figure 6. We can see that for this obstacle,  $b = (w + 2)(h + 2)$  and  $m = 2wh + 2(w + h) + 4$ , so that  $m(\mathcal{C}_{w,w})/b(\mathcal{C}_{w,w}) \rightarrow 2$  as  $w \rightarrow \infty$ . As for exterior obstacles, a long straight path quickly approaches our bound.

## 4 The Homing Robot

The Homing Robot is described in detail in [2]; we briefly describe its analogue in a hybrid grid here. The algorithm we present is identical to Wongngamnit

and Angluin’s algorithm when used in a pure occupancy grid. The robot is told the width  $w$  and height  $h$  of the connected hybrid grid  $\mathcal{G}$ , and it keeps track of its position relative to its start position with coordinates  $(H, V)$ , as well as the maximum and minimum of these values  $(H_{\max}, V_{\max})$  and  $(H_{\min}, V_{\min})$ . The robot’s goal is to visit all extremes of the boundary, so that it can calculate its absolute position; that is, the algorithm stops as soon as  $H_{\max} - H_{\min} + 1 = w$  and  $V_{\max} - V_{\min} + 1 = h$ . The Homing Robot thus uses only  $O(\log n)$  bits of memory.

The robot starts in some cell  $s$ . It checks the cells and lines at least as far north<sup>1</sup> as its current cell; if none of these are blocked, it goes one step north (this is *go-north* mode). Otherwise, it enters *deal-with-obstacle* mode:<sup>2</sup> in this mode, the robot turns right (that is, it turns so the obstacle is on its left) and goes around the obstacle until it is one move away from its start cell. It then returns to the closest northernmost cell and repeats the cycle. In this way it eventually reaches the north border of the boundary, at which point it goes around the boundary (in *deal-with-obstacle* mode) at most once until it has seen the extremes in all directions.

#### 4.1 The least upper bound

Let  $M$  be the number of moves the robot makes before it localizes. Wongngamnit and Angluin showed that  $M \leq 4.5n$  in [2] and were able to reduce this to  $4n$  in a subsequent paper; however, their lower bound for the worst-case only approached  $3n$ . We prove that this latter value is in fact the least upper bound for the Homing Robot, up to factors of  $o(n)$ .

**Theorem 6**  $M < 3n$ .

**PROOF.** First we find a bound for  $M$  that is similar to the one in [2]. Just as in [2], we know that the robot finishes *deal-with-obstacle* mode at least one cell farther north than it entered that mode. We also know that it does not enter any cell in both *go-north* mode and *deal-with-obstacle* mode. Let  $\mathcal{O}_1, \dots, \mathcal{O}_k$  be all of the obstacles that the robot has to deal with, and let  $\mathcal{O}_{k+1}, \dots, \mathcal{O}_f$  be the other obstacles (assume that the boundary obstacle is  $\mathcal{O}_1$ ). Thus we know that the robot is in *go-north* mode for at most  $h - k - 1$  moves. Let  $c_i$  be the number of moves that the circumnavigation robot takes to circumnavigate obstacle  $i$ .

<sup>1</sup> The Homing Robot described in [2] actually goes west instead of north and turns left instead of right.

<sup>2</sup> Known as *go-around* mode in [2]; I prefer *deal-with-obstacle* mode since the robot must do more than simply circumnavigate the obstacle.

**Lemma 7** *The Homing Robot is in deal-with-obstacle mode for obstacle  $i$  for at most  $\frac{3}{2}c_i - 2$  moves, where  $1 < i \leq k$ .*

**PROOF.** The robot always spends  $c_i - 1$  moves circumnavigating the obstacle, since it stops circumnavigating just before moving into its start cell. Suppose it is in cell  $s$  after its circumnavigation. Since the circumnavigation robot must make the same number of north as south and east as west moves to circumnavigate an obstacle, we know that  $c_i$  is even. Since the Homing Robot goes in the shorter direction to get to the closest northernmost point, we know that the second leg of its journey involves at most  $c_i/2$  moves; this must be a strict inequality, however, as follows. Suppose the robot spends another  $c_i/2$  moves going to the closest northernmost point. This will either be the northwest or the northeast corner; assume it is the former. So there are another  $c_i/2$  moves to go around the other direction to get back to  $s$ . But this return path must go through the northeast corner, so the northeast corner was closer after all, a contradiction. Adding the two quantities together, we find that the total number of moves spent dealing with this obstacle is at most  $\frac{3}{2}c_i - 2$ .

We know that the robot spends at most  $c_1 - 1$  moves in *deal-with-obstacle* mode on the boundary obstacle; we would like to somehow bound  $c_1$ . First,  $c_1 \geq 2w + 2h - 4$  (we subtract 4 since the boundary has at least four right turns — see Lemma 2);  $w \geq 1$ , so  $c_1 \geq 2h - 2$ . Thus  $c_1/2 \geq h - 1$ . Since the robot is always in *go-north* or *deal-with-obstacle* mode, we have finally that

$$\begin{aligned} M &\leq (h - k) - 1 + \frac{3}{2} \sum_{i=1}^k c_i - 2(k - 1) - (h - 1) - 1 \\ &= \frac{3}{2} \sum_{i=1}^k c_i - 3k + 1. \end{aligned} \tag{6}$$

Now we connect all of the obstacles with line barricades in order to use Theorem 4 to bound the sum. (This is where hybrid grids become necessary — if we connected the obstacles with full blocked cells, then we would reduce the number  $n$  of free cells and complicate things horribly.) Redefine  $m(s)$  for the multiple obstacle case to be the number of times the circumnavigation robot moves into cell  $s$  while circumnavigating *any* obstacle; note that Lemma 3 still holds. Let  $\mathcal{G}'$  denote any hybrid grid, and define  $m(\mathcal{G}') = \sum m(s)$  where the sum is over all free cells in  $\mathcal{G}'$ . Let  $\mathcal{G}_1 = \mathcal{G}$  be our starting grid (before we have connected any obstacles), so  $m(\mathcal{G}_1) = \sum_{i=1}^f c_i$ . Since the minimal obstacle is a single line, we know that  $c_i \geq 6$  for any  $i$ , so with  $p = f - k$ , we have that

$$m(\mathcal{G}_1) \geq \sum_{i=1}^k c_i + 6p \geq \sum_{i=1}^k c_i + 2p. \tag{7}$$

We now describe an inductive procedure for connecting an obstacle to the boundary. Suppose we have a grid  $\mathcal{G}_{j-1}$ . Construct  $\mathcal{G}_j$  from  $\mathcal{G}_{j-1}$  by first choosing any westernmost cell of any westernmost (interior) obstacle, and then drawing a line barricade due west until we hit the boundary (see Figure 7). So we have joined  $\mathcal{O}_i$  to the boundary  $\mathcal{O}_1$  for some  $i$ . Note that  $\mathcal{G}_f$  has only one (exterior) obstacle.

**Lemma 8**  $m(\mathcal{G}_j) \geq m(\mathcal{G}_{j-1}) - 2$ ; that is, we decrease the number of circumnavigation moves by at most two in connecting an obstacle.

**PROOF.** First, we have only changed  $m(s)$  for those cells  $s$  immediately to the north, south, east, and west of the line we added. The only way we could have *decreased* the total number of circumnavigation moves is if we have connected two disjoint barricades that were in the neighborhood of a single free cell; this can only happen when the line is one cell long (see Figure 7). In that case, we will have reduced  $m(s)$  for the two cells indicated in the diagram by exactly one each, without affecting any other cells.

In the worst case, then, we have lost  $2p$  moves in connecting the last  $p$  obstacles and  $2(k-1)$  moves in connecting the first  $k-1$  (we never connect the boundary to anything), i.e.

$$m(\mathcal{G}_f) \geq m(\mathcal{G}_1) - 2(k-1) - 2p \geq \sum_{i=1}^k c_i - 2(k-1), \quad (8)$$

where we obtained the last inequality from Equation 7. Thus  $\sum_{i=1}^k c_i \leq m(\mathcal{G}_f) + 2(k-1) < 2b + 2(k-1)$  by Theorem 4, where  $b$  is the number of border cells of the one obstacle in  $\mathcal{G}_f$ . But we know  $b \leq n$ , so we have  $\sum_{i=1}^k c_i < 2n + 2(k-1)$ . Substituting into equation 6, we find that

$$M \leq \frac{3}{2} \sum_{i=1}^k c_i - 3k + 1 < 3n + 3k - 3 - 3k + 1 < 3n. \quad (9)$$

## 5 Hexagonal grids

Results analogous to most of those so far presented hold for hexagonal grids as well. The proofs are nearly identical, so we only sketch the differences between the proofs for hexagonal grids and those for square grids in this section.

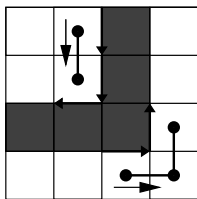


Fig. 4. When the arrows make a right turn, one fewer move is needed for the robot to keep even with them. When the arrows make a left turn, one more move is needed.

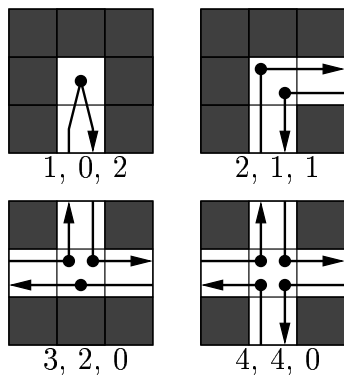


Fig. 5. The extremal cases in Theorem 4. The numbers are  $m(s), l(s), r(s)$ , from left to right.

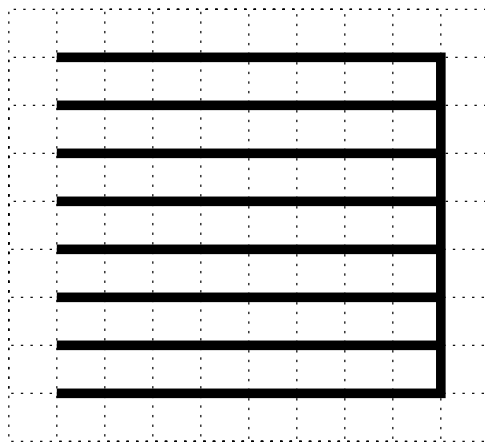


Fig. 6. The  $8 \times 7$  comb  $\mathcal{C}_{8,7}$ .



Fig. 7. We draw the center line obstacle to connect a westernmost obstacle (right) to the boundary (left).  $m(s)$  can only change for those cells marked with an asterisk, and those only because the line we drew is one unit long.

## 5.1 General properties

A *hexagonal hybrid grid* on a hexagonal tiling of the plane is defined in the same way as a square hybrid grid. Directions and coordinates must be handled slightly differently: if the grid is oriented as in Figure 8, then it makes sense to talk of the relative distance in the north direction between two cells, since the cells comprise well-defined rows. There are two other directions that have the same property, namely 10 o'clock and 2 o'clock; we choose the 2 o'clock direction and label it "east" (so 8 o'clock is "west"). Now we can again define the width and height of the grid. All of the other definitions in section 2.1 have immediate analogs, and the connectedness algorithm still works.

There is no simple result analogous to Lemma 2 — for instance, consider the obstacle of one cell, in which  $m = p$ , and the obstacle of two adjacent cells, in which  $m = p - 4$  (see Figure 8). The reason is that a straight path no longer borders a straight perimeter, so there is more variation in the number of perimeter lines that lie on a border cell.

Define  $m(s)$  as in Section 3.1. Lemma 3 still holds, with the same proof.

**Lemma 9**  $m(s)$  is the number of distinct connected barricades in the neighborhood of  $s$ .

The first result whose proof is slightly different than before is Theorem 4. Let  $\mathcal{O}$  be any interior or exterior obstacle in a hexagonal hybrid grid, and define  $m$  and  $b$  as before.

**Theorem 10**  $m < 2b$ .

**PROOF.** Let "a turn" be a sixty degree turn, so that the robot can make one or two turns to the left in one cell, and up to three right turns at once. Define  $l$  and  $r$  to be the total number of left and right turns, and define  $l(s)$  and  $r(s)$  to be the number of left and right turns in cell  $s$ . We thus know that  $r = l \pm 6$ . We proceed the same as before, balancing cells with  $m(s) > 2$  with cells with  $m(s) = 1$ . There are more cases to consider now, since  $m(s) \leq 6$  instead of 4, and it is possible to have a cell with  $m(s) = 3$  and  $r(s) = 1$ . Define  $m_1, \dots, m_6$  as in the proof of Theorem 4. Again we bound  $l$  and  $r$ ; see Figure 9 for the extremal cases.

- If  $m(s) = 1$  then  $r(s) \leq 3$  and  $l(s) \geq 0$ .
- Write  $m_2 = m_2^1 + m_2^2 + m_2^3$ , where  $m_2^1$  is the number of cells  $s$  with  $m(s) = 2$  and  $r(s) = 1$ ,  $m_2^2$  is the number of cells  $s$  with  $m(s) = 2$  and  $r(s) = 2$ , and  $m_2^3$  is the rest (with  $r(s) = 0$ ). If  $m(s) = 2$  and  $r(s) = 1$  then  $l(s) \geq 1$ ; if  $r(s) = 2$  then  $l(s) = 2$ , and otherwise  $l(s) \geq 0$ .

- Write  $m_3 = m_3^1 + m_3^2$ , where  $m_3^1$  is the number of cells  $s$  with  $m(s) = 3$  and  $r(s) = 1$ , and  $m_3^2$  is the rest. If  $m(s) = 3$  and  $r(s) = 1$  then  $l(s) = 4$ ; if  $r(s) = 0$  then  $l(s) \geq 3$ .
- If  $m(s) = 4$  then  $l(s) \geq 6$ .
- If  $m(s) = 5$  then  $l(s) \geq 9$ .
- If  $m(s) = 6$  then  $l(s) = 12$ .

Now we can bound  $l$  and  $r$ .

$$\begin{aligned}
l &\geq m_2^1 + 2m_2^2 + 4m_3^1 + 3m_3^2 + 6m_4 \\
&\quad + 9m_5 + 12m_6 \\
r &\leq 3m_1 + m_2^1 + 2m_2^2 + m_3^1 \\
\implies l - r &\geq -3m_1 + 3m_3 + 6m_4 + 9m_5 + 12m_6.
\end{aligned} \tag{10}$$

As in Theorem 4, we can now show that for either type of obstacle,

$$m_1 - m_3 - 2m_4 - 3m_5 - 4m_6 - 2 \geq 0. \tag{11}$$

The differences are that now  $l - r = \pm 6$  and that there are six different directions in which to find the extra left turns. Again we know that  $m = \sum_{i=1}^6 im_i$  and  $b = \sum_{i=1}^6 m_i$ , so

$$m \leq \sum_{i=1}^6 2m_i - 2 = 2b - 2 < 2b \tag{12}$$

Again this is a least upper bound, as we can construct a comb-like obstacle and a long straight path in a hexagonal grid.

## 5.2 The Homing Robot in a hexagonal grid

The Homing Robot has a natural generalization to a hexagonal grid. It is slightly more difficult to head due north, since the robot can only travel in odd clock directions (i.e. 1 o'clock, 3 o'clock, etc.); therefore the robot does not head due north and instead heads due one o'clock in *go-north* mode. In this way the robot still ends up one unit farther north after each such move. The *deal-with-obstacle* mode is implemented in the same way as in the square case. Let  $\mathcal{G}$  be our connected hexagonal hybrid grid, and let  $M$  be the number of moves for the Homing Robot to localize in  $\mathcal{G}$ . Let  $n$  be the number of free cells in  $\mathcal{G}$ .

**Theorem 11**  $M < 3n$ .

**PROOF.** Let  $w$  and  $h$  be the width and height of the grid (in the east and north directions, respectively). Define  $\mathcal{O}_i$ ,  $c_i$ ,  $k$ ,  $p$ , and  $f$  as before. By the same reasoning as in the proof of Theorem 6, we know

$$\begin{aligned} M &\leq (h - k) - 1 + \frac{3}{2} \sum_{i=1}^k c_i - 2(k - 1) - (h - 1) - 1 \\ &= \frac{3}{2} \sum_{i=1}^k c_i - 3k + 1. \end{aligned} \tag{13}$$

We again proceed as in Theorem 6; the next difference arises when connecting obstacles to the boundary. The procedure is nearly the same, in that we choose a westernmost point of a westernmost obstacle and draw a line due west; only now the line is very wrinkled (see Figure 10). We still decrease the number of moves if we connect two distinct barricades in the neighborhood of one cell, so there are at most two cells  $s$  which have  $m(s)$  decreased by at most one. So again we have  $\sum_{i=1}^k c_i \leq m(\mathcal{G}_f) + 2(k - 1) < 2b + 2(k - 1)$ , and substituting into equation 13, we obtain  $M < 3n$ .

Theorem 11 is a least upper bound, as a hexagonal analog of the worst-case example given in [2] can be easily constructed.

There is no reason to stop with square and hexagonal grid tilings; indeed, it is hoped that these techniques will perhaps aid in finding bounds for other localization problems, maybe involving triangular grids or arbitrary planar graphs (a similar problem has been treated in [1]). To the knowledge of the author, these problems are still open.

## 6 Acknowledgements

This paper was completed as a project for the Research Experience for Undergraduates in Duluth, Minnesota, run by Dr. Joseph A. Gallian. Many thanks to the participants in the program, and special thanks go to Jan Siwanowicz for his help in proving Theorem 4. Supported by NSF grant DMS-9820179 and NSA grant 904-00-1-0026. I also thank the referee who suggested the triangular grid and planar graph problems.

## References

- [1] P. Bose, A. Brodnik, S. Carlsson, E. Demaine, R. Fleischer, L. Jacobsen, A. Lopez-Ortiz, P. Morin, and J. Munro, Online routing in convex subdivisions,

In 11th Annual International Symposium on Algorithms and Computation (ISAAC '00), 2000.

- [2] C. Wongngamnit and D. Angluin, Robot localization in a grid, *Information Processing Letters* 77 (2001) 261-267.
- [3] C. Wongngamnit and D. Angluin, The robot, the grid, and the algorithm, Technical Report YALE/DCS/TR-1188, Yale University Computer Science Department, New Haven, CT, 1999.

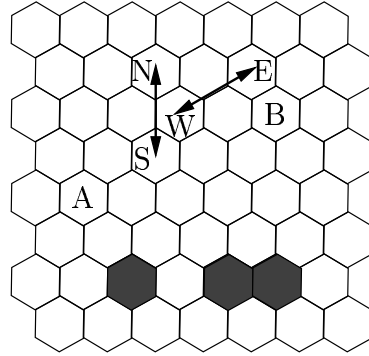


Fig. 8. A hexagonal (hybrid) grid and the cardinal directions, along with two obstacles. The coordinates work as follows: if cell  $A$  has coordinates  $(0, 0)$ , then  $B$  is at  $(5, 2)$ , that is, it is five rows east and two rows north of cell  $A$ .

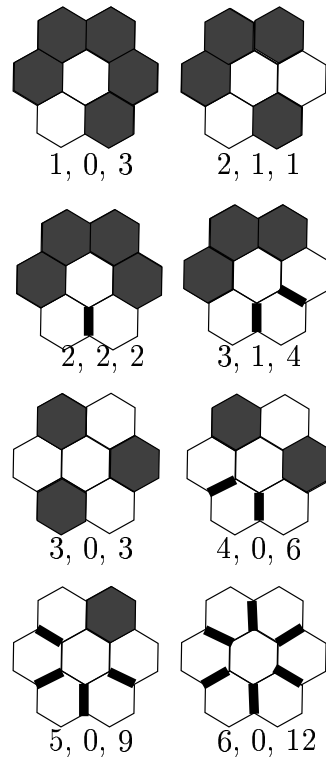


Fig. 9. The extremal cases in the proof of Theorem 10. The numbers are  $m(s), l(s), r(s)$ .

