Concepts

Measurement outcomes

A_z set of all possible measurement outcomes $z'$with respect to the grid cell C of all laser scans that are recorded close enough to potentially measure the cell. These measurement outcomes will be used for computing the entropy and the mutual information between the map and each cell.

Measurement outcomes $z'$is a set of three values obtained from the proposed inverse measurement model $[l_{free}, l_{occ}, l_0]$. After a new laser measurement is received, all the cells that can be seen are found, then we apply the inverse measurement model to obtain the set of three values stated above.

The number of measurement outcomes is given by the number of ways to sample k elements from the set of 3 laser measurement outcomes $[l_{free}, l_{occ}, l_0]$with replacement and disregarding different ordering. For calculating the number of combinations use the equation (3.13 from the Thesis or 12 from the paper). The explanation in the paper is better for this point.

Outcomes probabilities

Algorithm 1 computes the probabilities of the outcomes by propagating the probability mass through the graph, for this, the algorithm uses a hash table P(<-, -, ->) which is indexed by the measurement histograms.

Measurement histograms

This one is the possible combinations based on all the available measurements, the order is not important in this case.

Notes

The outcome probabilities from my understanding are the probabilities produced by propagating the probabilities of the current reading to the next one (Even if there are no more readings), for example we have two measurements that saw a cell (Level 2), then we will propagate the probabilities for the next level (Which is proposed by the algorithm). Looking at the algorithm it starts in one of the cells that fulfills the condition f + o + u = r, and from that point we calculate the probability of the next level based on the current measurement outcomes and probabilities. See the left hand side of the **Figure 3.4**, probabilities are calculated in that way; however in the right hand side we have the representation of all the measurement outcomes z'(In this case replacement takes place so we have a reduction in the number of combinations, this is equivalent to different combinations).

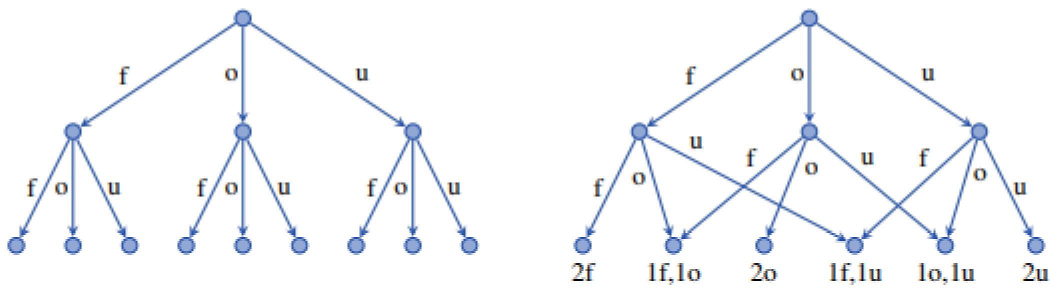

Figure 3.4: Space of measurement outcomes of a grid cell given k laser scans, where $k = 2$. Here, "free" is denoted by f, "occupied" by o, and "unknown" by u. Left: In general, there are 3^k measurement outcomes, which is exponential in k. Right: The number of measurement outcomes is quadratic in k when exploiting the standard measurement model.

Algorithm test

This is my interpretation.

Testing with a set of two measurements (k = 2) for a specific cell (The cell is not relevant at all)

<f, o, u> that fulfills. f + o + u = 1
Using 3.13 we get 3 (Three possible different results)
Using 3.14 we get 9 (All possible probabilities)
<1, 0, 0>, <0, 1, 0>, <0, 0, 1>

Loop for r = 1
Taking: <1, 0, 0>
P(<2, 0, 0>) += P(<1, 0, 0>) * P(Z_i^jobserves C as free);
P(<1, 1, 0>) += P(<1, 0, 0>) * P(Z_i^jobserves C as occupied); - Adding
P(<1, 0, 1>) += P(<1, 0, 0>) * P(Z_i^jdoes not observe C); - Adding

Taking: <0, 1, 0>
P(<1, 1, 0>) += P(<0, 1, 0>) * P(Z_i^jobserves C as free); - Adding
P(<0, 2, 0>) += P(<0, 1, 0>) * P(Z_i^jobserves C as occupied);
P(<0, 1, 1>) += P(<0, 1, 0>) * P(Z_i^jdoes not observe C); - Adding

Taking: <0, 0, 1>
P(<1, 0, 1>) += P(<0, 0, 1>) * P(Z_i^jobserves C as free); - Adding
P(<0, 1, 1>) += P(<0, 0, 1>) * P(Z_i^jobserves C as occupied); - Adding
P(<0, 0, 2>) += P(<0, 0, 1>) * P(Z_i^jdoes not observe C);

<f, o, u> that fulfills. f + o + u = 2
Using 3.13 we get 6 (Three possible different results). These were filled before in the loop when r = 1
Using 3.14 we get 36 (All possible probabilities) - There is an incongruity in this equation. On the left side we got 27 and on the right side of the equation we got 36). Also note that the 36 is the total computation of probabilities including the ones at the first step.
<2, 0, 0>, <1, 1, 0>, <0, 2, 0>, <0, 1, 1>, <1, 0, 1>, <0, 0, 2>

Loop for r = 2
Using 3.13 we get 6 (Six possible different results). T
Taking: <2, 0, 0>

P(<3, 0, 0>) += P(<2, 0, 0>) * P(Z_i^jobserves C as free); - 1

P(<2, 1, 0>) += P(<2, 0, 0>) * P(Z_i^jobserves C as occupied); - Adding - 2

P(<2, 0, 1>) += P(<2, 0, 0>) * P(Z_i^jdoes not observe C); - Adding - 3

Taking: <1, 1, 0>

P(<2, 1, 0>) += P(<1, 1, 0>) * P(Z_i^jobserves C as free); - Adding

P(<1, 2, 0>) += P(<1, 1, 0>) * P(Z_i^jobserves C as occupied); - Adding - 4

P(<1, 1, 1>) += P(<1, 1, 0>) * P(Z_i^jdoes not observe C); - Adding - 5

Taking: <0, 2, 0>

P(<1, 2, 0>) += P(<0, 2, 0>) * P(Z_i^jobserves C as free); - Adding

P(<0, 3, 0>) += P(<0, 2, 0>) * P(Z_i^jobserves C as occupied); - 6

P(<0, 2, 1>) += P(<0, 2, 0>) * P(Z_i^jdoes not observe C); - Adding - 7

Taking: <0, 1, 1>

P(<1, 1, 1>) += P(<0, 1, 1>) * P(Z_i^jobserves C as free); - Adding

P(<0, 2, 1>) += P(<0, 1, 1>) * P(Z_i^jobserves C as occupied); - Adding

P(<0, 1, 2>) += P(<0, 1, 1>) * P(Z_i^jdoes not observe C); - Adding - 8

Taking: <1, 0, 1>

P(<2, 0, 1>) += P(<1, 0, 1>) * P(Z_i^jobserves C as free); - Adding

P(<1, 1, 1>) += P(<1, 0, 1>) * P(Z_i^jobserves C as occupied); - Adding

P(<1, 0, 2>) += P(<1, 0, 1>) * P(Z_i^jdoes not observe C); - Adding - 9

Taking: <0, 0, 2>

P(<1, 0, 2>) += P(<0, 0, 2>) * P(Z_i^jobserves C as free); - Adding

P(<0, 1, 2>) += P(<0, 0, 2>) * P(Z_i^jobserves C as occupied); - Adding

P(<0, 0, 3>) += P(<0, 0, 2>) * P(Z_i^jdoes not observe C); - 10

Algorithm
Algorithm input: Set of **k** laser measurements **Z**, cell **C**
Algorithm output: Probabilities P(<-, -, ->) of the outcomes (free, occupied, not observed)
Initial condition: P(<0, 0, 0>) = 1 (This is the root of the graph)
Loop
for r=1 .. k **do**
    for all <f, o, u> with f + o + u == r **do**
        P(<f + 1, o , u>) += P(<f, u, o>)*P(Z_i^jobserves C as free);
        P(<f, o + 1 , u>) += P(<f, u, o>)*P(Z_i^jobserves C as occupied);
        P(<f , o , u + 1>) += P(<f, u, o>)*P(Z_i^jdoes not observe C);
    end for
end for
return P(<-, -, ->)