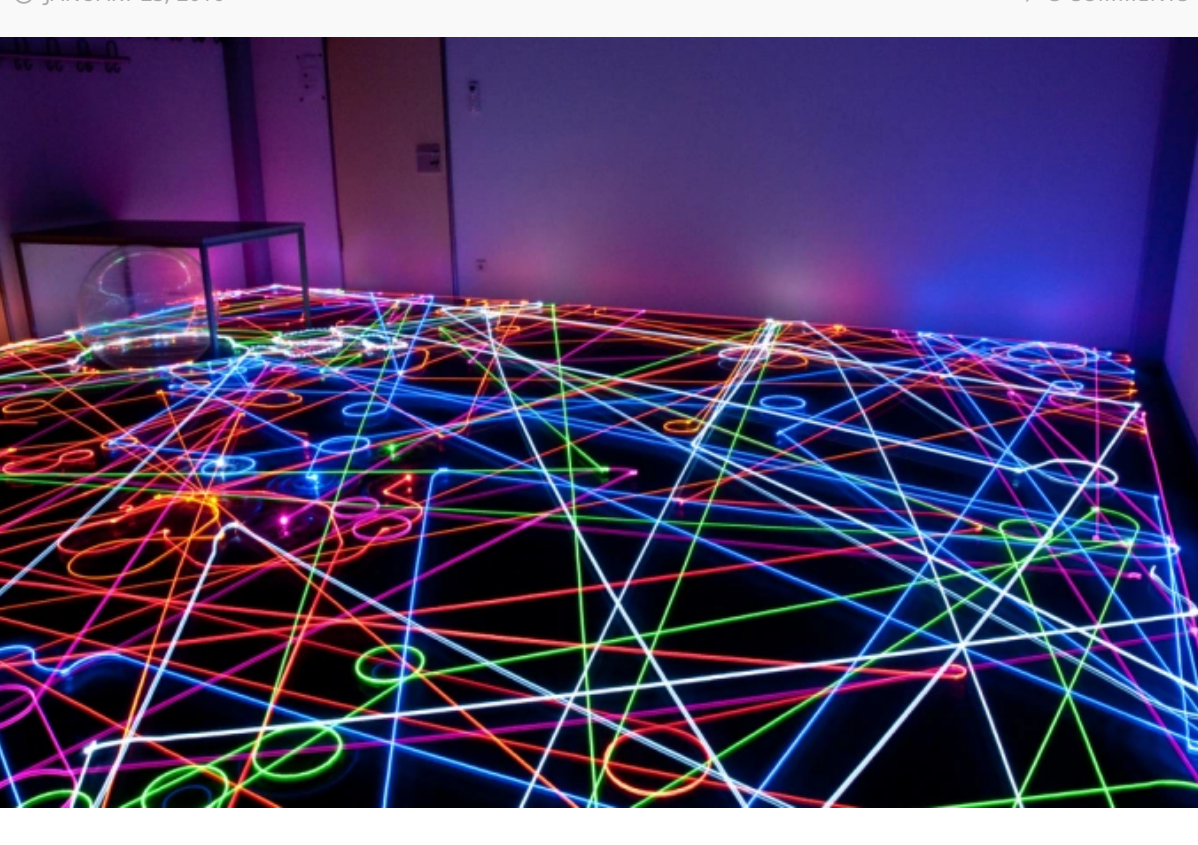
Code and a glass of wine

Roomba algorithms and visualization

JANUARY 25, 2016 3 COMMENTS



I once had an interview question asking for an algorithm for a Roomba that ensures it covers every square of a room divided into grid cells, given that the room shape and location of obstacles are unknown. It's similar to the idea of solving a maze, except that instead of getting to a specific point, you're trying to visit every point in the room – to clean it!

It's a pretty common problem, but I hadn't seen it in the guise of a physical robot before. Running a Depth First Search covers every piece of floor easily enough, but casting it as a physical device that has to move implies a large cost to popping back up the stack that's generated during DFS. There's a lot of backtracking in a DFS based approach for a Roomba, so it makes for a slower vacuuming job.

It made me wonder whether there was some better approach than DFS that would be more efficient.

The way of the Ox

I came across this question on the Robotics StackExchange. The answer from Josh Vander Hook mentions the concept of a Boustrophedon path, which roughly means the "way of the ox" – the typical back and forth sweep of a farmer ploughing a field. When forced to mow lawns as a child to earn my keep, I never realised my approach had such a long and unpronounceable name that would one day end up on Wikipedia.

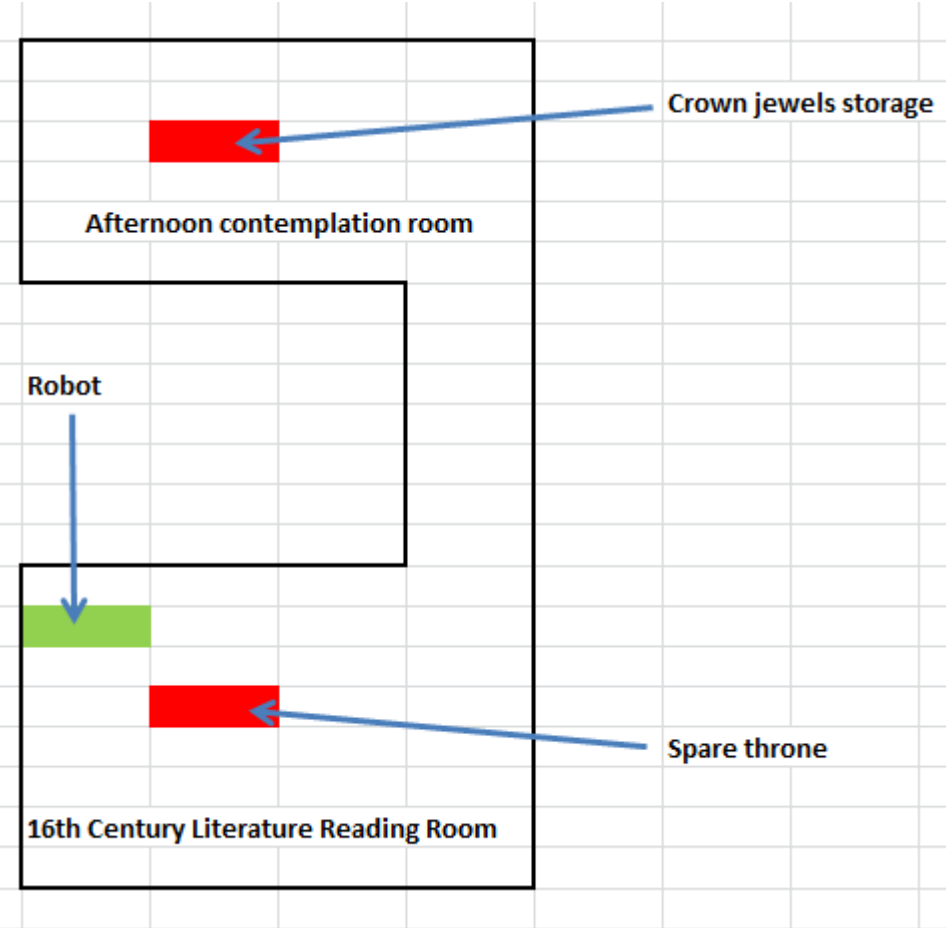
The Roomba doesn't know the room it's cleaning, and it doesn't build a map as it goes, either. Without knowing the room layout there's no way to guarantee a particular approach will be efficient. A good approach might be to find an edge from the starting point as soon as possible, and then whip out the Boustrophedon. However, all the back and forth sweeps will double over the initial path taken to find an edge. Navigating to the closest wall will be most efficient, but the robot doesn't know where the closest wall is.

My kingdom for a map

I was curious how much difference having a complete map would make. In reality rooms and obstacles aren't static, so a map may not be accurate. But let's say we're talking about cleaning some disused wing of Buckingham Palace, where nothing ever moves and no one ever goes. But the Queen is impatient, so the robot has to be fast.

If we had a complete map, would it allow an efficient algorithm to determine the fastest path for a vacuuming robot to take? It sounds similar to the Travelling Salesman problem – finding the shortest path that visits each point on a map once only. This is different in a couple of ways – it's ok to re-vacuum a spot, and the robot can only travel direct to a cell if it's currently next to it, otherwise it vacuums over other cells on the way.

Thinking about Buckingham Palace, here's a ~~shameful abomination~~ piece of art that I ~~knocked up in Excel~~ crafted in Photoshop to illustrate the problem:



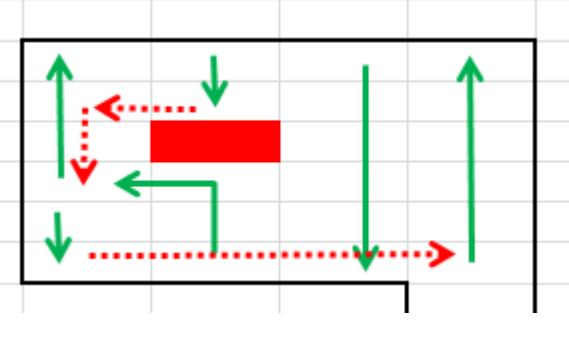
Basic Depth First Search

Take a standard Depth First Search, and have the robot search for options in the order of Up, Right, Down, Left. With the robot starting on the green cell it will work its way into the corridor, then go up to the Afternoon Contemplation Room. From there, it will zig-zag up and down towards the left of the room until it finally finishes the room, working around the crown jewels on the way.

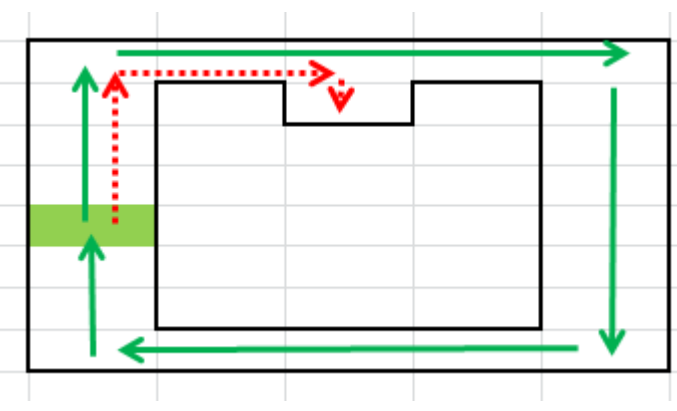
The problem is that the robot has to backtrack through the entire room to get out of it after it's done there. It'd be much faster to just head straight back to the right until reaching the corridor, but that's not how a DFS based approach works.

Greed is better

A greedy algorithm approach would choose to visit the nearest cell that's not been covered, getting there by the fastest route. Assuming the same order of Up, Right, Down, Left, it would arrive into the room through the corridor in the bottom right, zig-zag through, and then take a shortcut back out:



The backtracking paths are shown in red – DFS would have backtracked through the entire room! So this approach is better than DFS, but is it the optimal algorithm? What about a scenario like this?



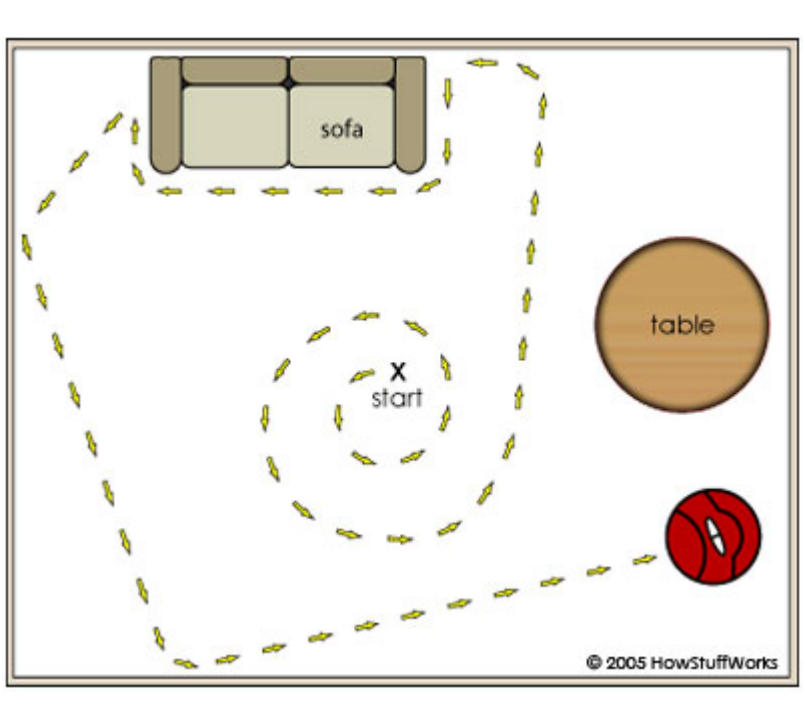
The best path is to cover that little nook along the top as the robot goes by, which would incur a single cell backtrack into the corridor again. But the greedy approach just looks to make the best decision on the spot without analysing the future implications of that choice, so it skips straight past the nook without realising it will cost a lot later. A more sophisticated algorithm could handle this scenario, but there will likely be other scenarios that are still sub-optimal.

Using the map should allow generating a route with a much better guarantee of performane, but the question is whether there's a feasible/scalable approach. The Travelling Salesman Problem is NP Hard, but I thought the variations in this problem might change that. A bit of research and reading through Stanford lecture notes seems to imply that's not the case – this problem is still NP Hard. However, there are approaches that could lead to an approximation that's at most twice the shortest path.

How a Roomba actually works

Kyle's answer on Robotics SE includes an excerpt from an interview with someone from iRobot (the makers of the Roomba). The approach for the Roomba is to spiral until it finds a wall/obstacle, and then attempt to sweep back and forth. Spiralling is an efficient way to cover ground when you don't know how big the space is, and it also means the Roomba will find the nearest wall first.

There's a link to How Stuff Works, with a diagram showing what they observed of the Roomba's behaviour:



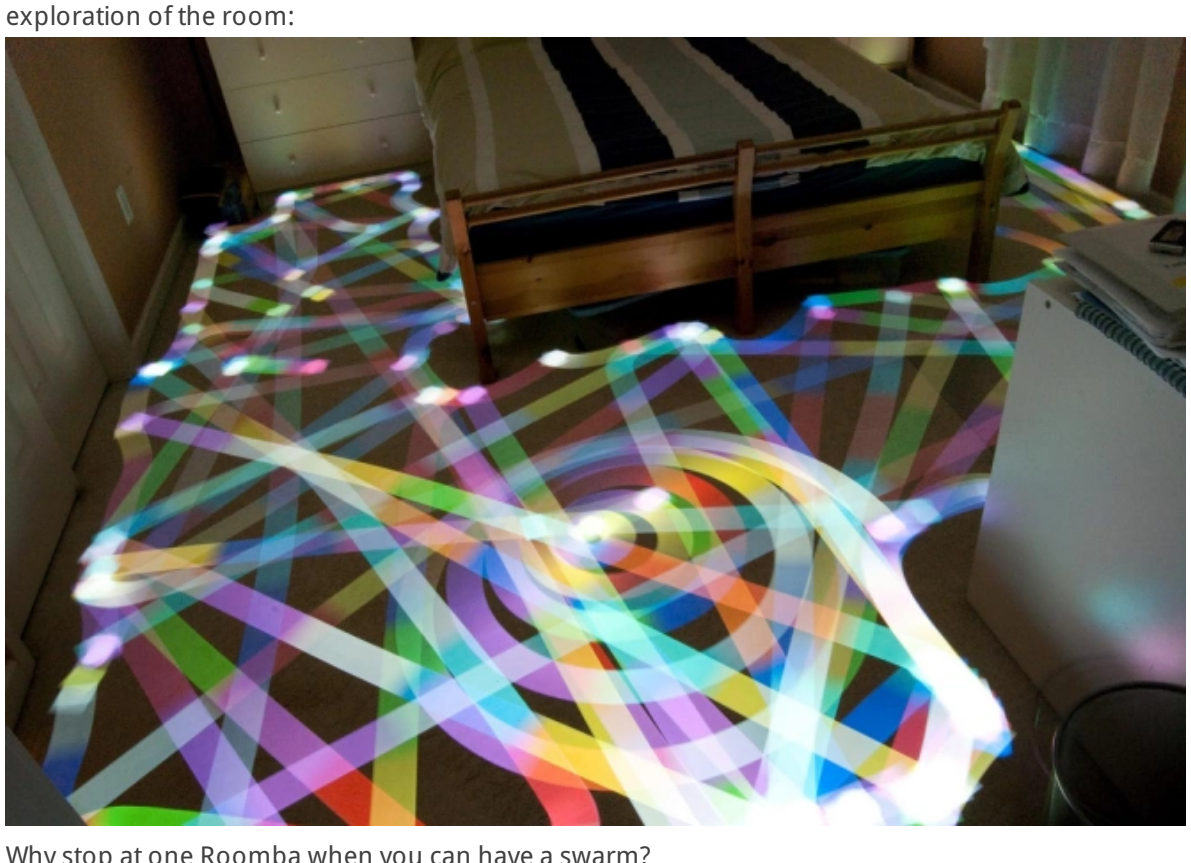
Braava does use a map

Roomba doesn't use a map, but its companion product does. Braava uses "NorthStar" cubes placed around the house that help it map the room as it goes. Presumably it uses the map to aid in backtracking efficiently when it needs to, and possibly even combines the map with heuristics about common room shapes to come up with good approaches. Rather than spiralling, it dissects the room into segments, and sweeps each segment. More details on the site.

The Roomba as a tool of art

All of this is very good and well, but it's missing the true purpose of the Roomba, which is to ferry around LEDs in a dark room to create cool patterns in long exposure photos. Check out the flickr group dedicated to it.

Here you can see the spiralling of the Roomba, followed by what looks like a rather drunken exploration of the room:



Why stop at one Roomba when you can have a swarm?

And of course, it wouldn't be complete without turning the Roomba into a Spirograph:

After all that talk about efficiency, really the best algorithm would be the one that makes the coolest shapes in your room.

Share this:

Twitter Facebook

Like

One blogger likes this.

Related

Detecting spikes in time series with Reactive Extensions
August 8, 2016
In "algorithms"

Solving GCHQ's Christmas nonogram in 0.07 seconds
March 9, 2016
In "algorithms"

NDC Sydney talk
October 25, 2016
In "conferences"

ALGORITHMS, VISUALIZATION ALGORITHMS, VISUALIZATION

← SOLVING BOGGLE BOARDS AT SCALE

IDENTIFYING MARKET SPOOFING WITH DATA VISUALIZATIONS →

3 thoughts on "Roomba algorithms and visualization"

Robotic Vacs
April 22, 2016 at 12:09 pm

As a robotics enthusiast, a programmer, and a lover of robotic vacuums I found this article great! If the literature at iRobot's website is correct, I believe that the Roomba 980 does use a map in order to determine the most efficient path to clean.

Great article, and it's cool to see Roomba, and DFS, and the Traveling Salesman problem all mentioned in the same place.

Like

REPLY

Niall Connaughton
April 23, 2016 at 12:13 am

Thanks for your comment! Yes, it's interesting to wonder how much of the formalities of Travelling Salesman the people at iRobot went through to design their software, or if it was more experimental and empirical, given the varying complexities the robots would have to deal with (furniture, stairs, rugs, ramps, surface types, pets, etc). I like to think they had a series of test rooms that they ran an army of robots through to verify different approaches, and each week one of them got kicked off the show.

Like

REPLY

Hafidz Jazuli L
November 25, 2017 at 6:21 pm

wow…
🙂

Like

REPLY

Leave a Reply

Enter your comment here...

Recent Posts

Canberra Wine District

Wine Tasting Around the World

Wine tasting notes – you couldn't make them up… could you?

NDC Sydney talk

Twitter Analysis: Hillary and Obama's Convention Speeches

Categories

.net

algorithms

C#

conferences

coursera

data

public speaking

Reactive Extensions

rxjs

Uncategorized

visualization

wine

Archives

January 2019 (2)

April 2017 (1)

October 2016 (1)

August 2016 (4)

May 2016 (1)

March 2016 (3)

January 2016 (1)

December 2015 (1)

November 2015 (1)

August 2015 (1)

July 2015 (1)

May 2015 (2)

January 2015 (1)

RSS

Register

Log in

Entries feed

Comments feed

WordPress.com

Follow Blog via Email

Enter your email address to follow this blog and receive notifications of new posts by email.

Enter your email address

FOLLOW