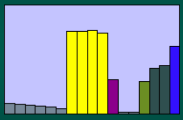
Daniel Keer’s account of “snapshot following”

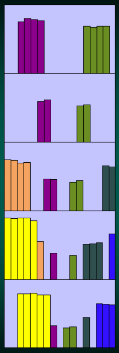
How the Snapshot Following Works

THE PROBLEM:

To compare a route of snapshots in memory…

…with what the ant can currently see.

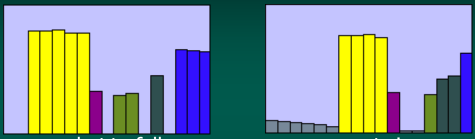




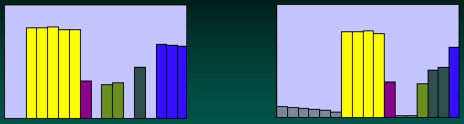
…and come up with an estimate of the **distance left to move** and the **direction** to turn to move the ant back to the location where the snapshot was takem..

How the Snapshot Following Works

At each step, the ant will compare the current view with the current snapshot it is following…

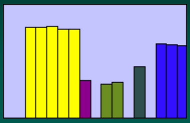


…to see how it needs to move to line up the current view with the remembered snapshot view.



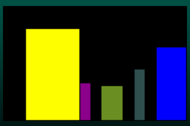
- It is obvious to the human eye that the ant needs to turn approx. 4 bar-widths to the right to get the yellow landmark to match up.
- You may also see that the yellow object in view is narrower and shorter than the snapshot, giving an indication of a longer current distance away than that in the snapshot.
- But how can the ant determine this?

The solution I implemented first processes the ‘raw visual data’ – coloured bars in the snapshot…



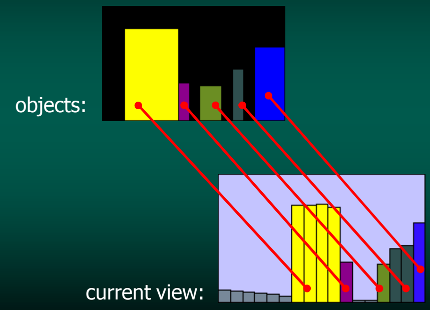
```
[ [“empty”, 0], [“empty”, 0], [“yellow”, 6.268],
  [“yellow”, 6.317], [“yellow”, 6.170], [“yellow”, 6.175],
  [“purple”, 2.634],   [“empty”, 0],   [“green”, 2.370],
  [“green”, 2.464],   [“empty”, 0],  [“darkgrey”,3.571],
  [“empty”, 0],  [“blue”, 5.170], [“blue”, 5.101],
  [“blue”, 4.999]  ]
```

…into a set of perceived coloured objects.

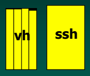


```
[ [“yellow”, 6.256, 3, 7],
  [“blue”, 5.090, 15, 17],
  [“green”, 2.417, 10, 11],
  [“darkgrey”, 3.571, 13, 13],
  [“purple”, 2.634, 8, 8]    ]
```

Each of these objects are then compared with what can be seen in the current view. Matches are made primarily by colour:



Then, for each object the ant has identified…



The average height is compared to find a **percentage match for height**:

(viewed height / snapshot height)

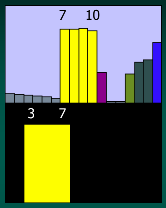
In this case, the percentage match is 0.951, indicating that the object is further away than it was when the snapshot was taken

The width is also compared:

(viewed width / snapshot width)

In this case, the **percentage width match** is 0.8, again indicating that the object is further away than in the snapshot.

(For Each Object)



An indication of the **turn** needed is calculated:

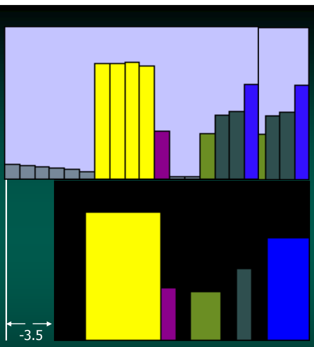
½ (start_point_in_snapshot – start_point_in_view
+ end_point_in_snapshot – end_point_in_view)

i.e. the combined difference between the respective start and end points of the object

In this case:

½ ((3 - 7) + (7 – 10)) = -3.5

(a turn to the right of 3.5 bar widths)



Which, if enacted, would line the object up perfectly.

When this has been completed *for each object*, there will be corresponding values for:

- Width percentage match
- Height percentage match
- Recommended turn

These values are then averaged for all the objects in view to get the overall values for the snapshot.

Using these overall measures of how well the snapshot matches, the ant then uses a set of simple rules to navigate:

- If the snapshot appears to be a good match (in terms of width and height), the ant will follow it. It will perform any turn actions necessary, and then take a step.
- When the height match is very close to 1 (indicating that the correct distance has been reached), the ant will start following the next snapshot in the trail.
- If the percentage match gets too low, the ant will assume it has lost its way, and revert into random search mode.

There is slightly more to the snapshot following procedure than has been detailed here, but I think it is evident that the fundamentals are fairly simple.

Ants only have fairly simple brains, so any innate behaviour will not be that complicated in real life.

The point to be made is that complicated intelligence is not necessary for complicated behaviour to arise.