

Continuation of Topological Mapping by Gap Sensing

November 16, 2005

Problem

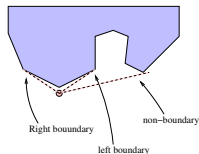
Define an algorithm for a mobile robot to build a topological map of a polygonal world

- Don't use odometry
- World contains polygonal obstacles and may have a polygonal boundary
- For now restrict to convex obstacles

Approach

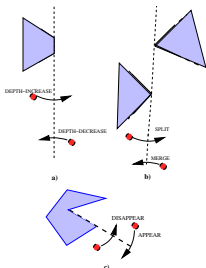
Use the gap sensing paradigm (Tovar et. al.)

- *Gap sensor* an idealized range sensor, reports discontinuous changes in depth at the boundary (*gaps*)
- Gap hides some portion of the world
- Robot uses the gap sensor to track the changes in the set of gaps as it moves



Gap events

- Occur when the extended portion of a bitangent is crossed
- Can be used to find edges in the visibility graph



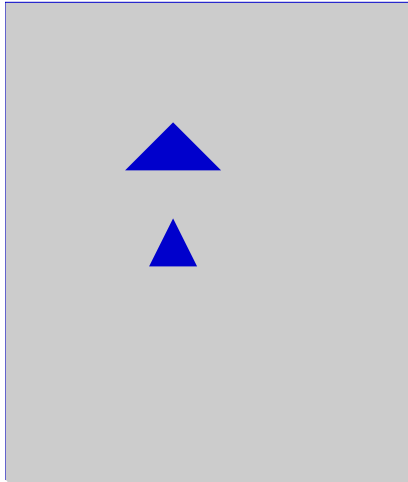
Robot

- Has a *metric* gap sensor
- Can follow a boundary at some ϵ distance
- Can drop a marker near a boundary and recover it upon return
- Can move toward a gap

Algorithm

1. while unvisited vertices exist ,
2. reach vertex by gap chasing
3. drop marker
4. follow boundary,
5. Until(found-marker())
 # adds information to the map.
6. handle-gap-events()
7. when vertex reached, record neighbors
8. merge start with finish
9. enforce constraints imposed by this equivalence
 to match other equivalent nodes

Extended Example

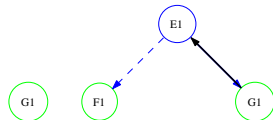




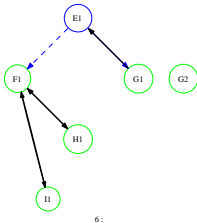
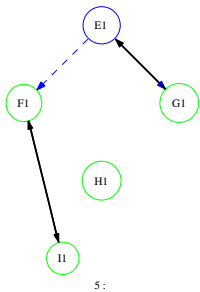
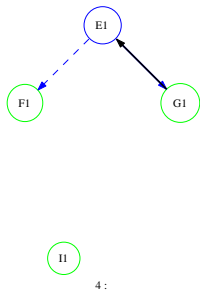
1 : --Initial--

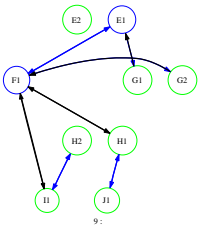
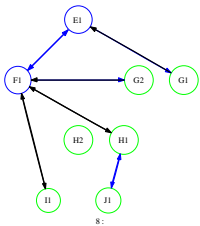
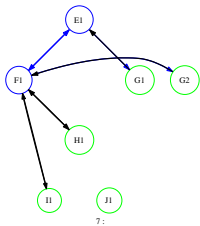


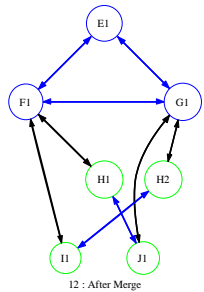
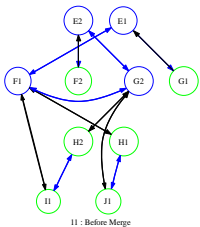
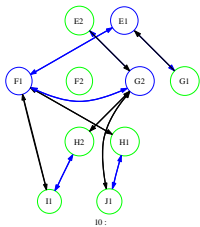
2:

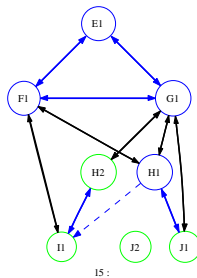
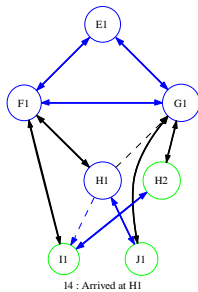
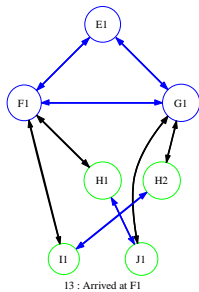


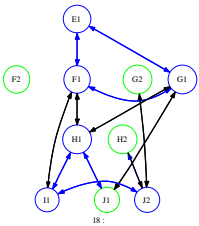
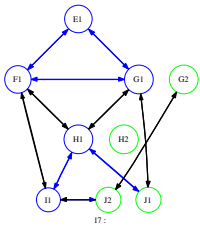
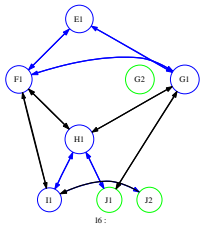
3 : Arrived at E1

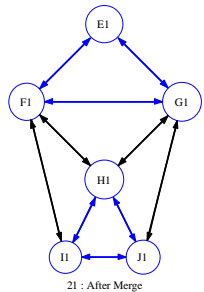
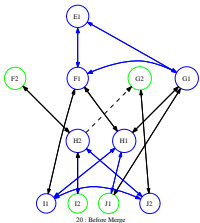
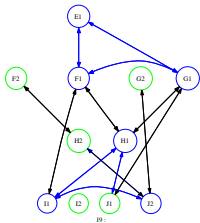








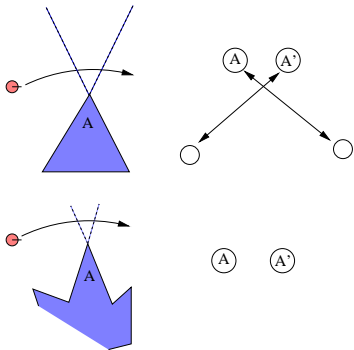




Duplicate Nodes

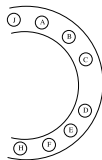
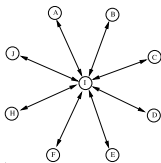
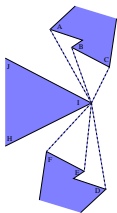
Two occasions where duplicate nodes are created

- At the end of a loop
- Due to depth changes



Neighborhoods

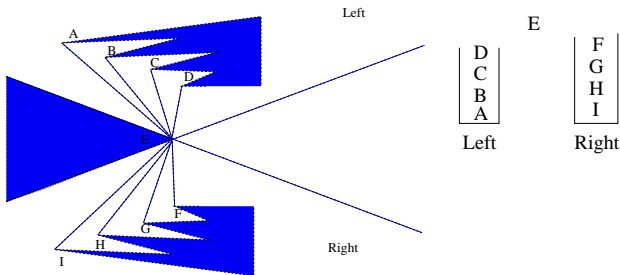
- Each vertex has a set of neighbors ordered with respect to its boundary
- In order for the recursive merge algorithm to work correctly, we must correctly match orderings of equivalent nodes.



Partial Orderings

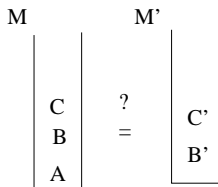
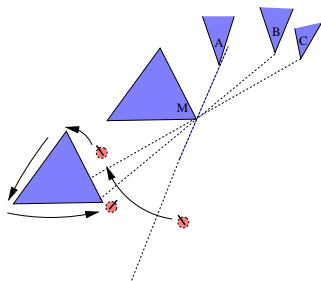
How do we maintain a partial ordering on the set of neighbors of an unvisited node?

- Use a FILO structure
- Prevents creation of duplicate nodes
- Also maintains a partial ordering on neighbors



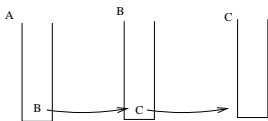
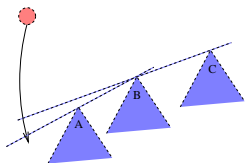
Merging Nodes

- Combine nodes with partial orderings

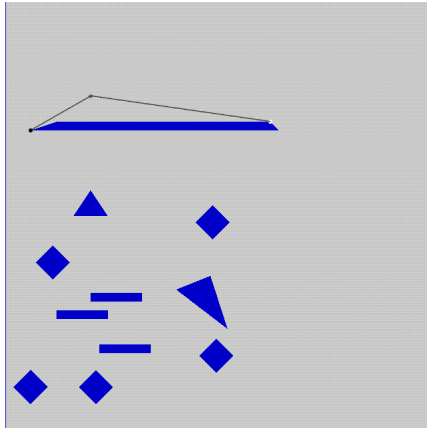


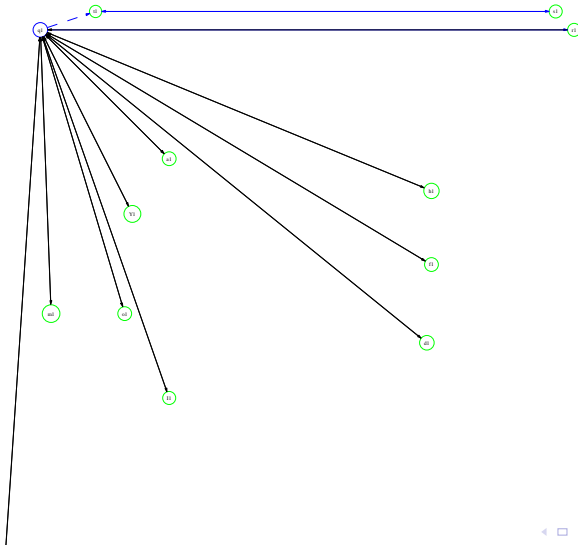
A Partial Solution

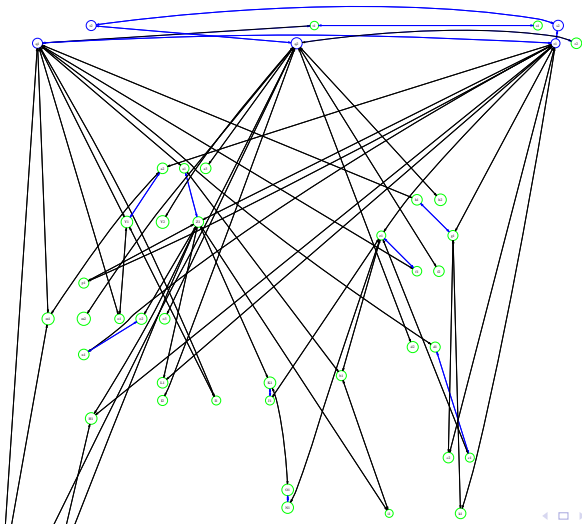
- Use pointers to maintain the correct settings
- Provides a local map back to any obscured feature

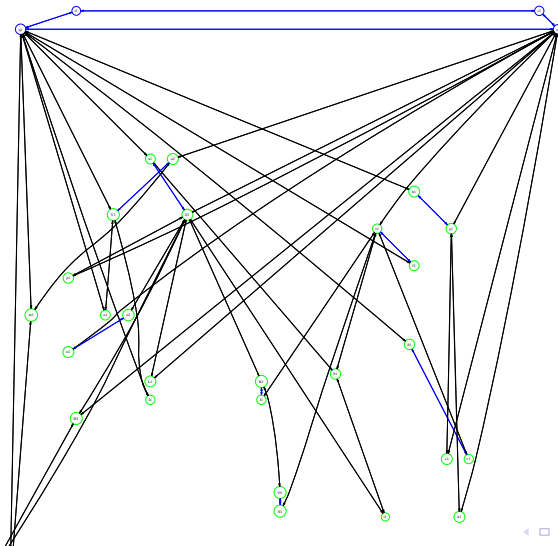


Example 2









Exploration Order

- Need to ensure that after circling all obstacles no duplicates exist
- Computation is dominated by shortest path computations but exploration order may be more important
- What exploration strategies cause the algorithm not to terminate?

Exploration Order (cont.)

- Information is limited
- Ideally, maintain a that shortest path to visited nodes goes through visited other nodes
- Not always possible

Conclusions

- Need to demonstrate completeness and correctness for convex obstacles before moving on to non-convex obstacles
- We continue to study the problem of exploration strategy
- Greedy strategy seems to be the best choice but this is not yet justified