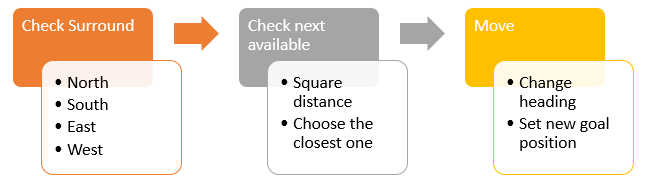
**Exploration Algorithm**

According to our goal, the algorithm must ensure the coverage of the entire scenario. To accomplish this objective, we need to communicate with the mapping task to retrieve the available information, and take decision based on that knowledge.

For practical purposes we assume that the kinematics and the mapping task are working perfectly, therefore, the algorithm can make high level decisions to reach the goal of cover/visit the scenario as much as possible.

The followed process is illustrated in the figure XXX, there are 3 main steps, that allow to obtain a new goal position each time that a cell is already explored.



We defined 5 cell states, that will allow us to track the coverage of the map.

* OBSTACLE (O): The IR sensors have detected an obstacle in their reliable range. Priority 1
* LINE (L): The line sensor has detected a line in the floor. Priority 2
* BORDER (B): Virtual border put around an obstacle to give free space to avoid clashing with the physical object. Priority 3
* VISITED (V): The robot has been over that cell. Priority 4
* EXPLORED (E) = The robot has detected an empty space using its IR sensors, that means that is available for visiting but it not visited yet. Priority 5
* UKNOWN (U) = Unexplored cell, inside the external limit of the map. Priority 6

They have different priority to avoid overwrite a mapped cell, being 1 the highest priority. For instance, a VISITED cell, cannot be labeled as EXPLORED again, regardless if the sensors detect that there is an empty space, because VISITED has higher priority than EXPLORED.

**Check Surround:** In this step, we retrieve information from the map about the status of the surrounding cells.

**Check next available:** Based on the information from the previous step we make decision follow the rules stated bellow.

* **IF** the NORTH cell is EXPLORED **THEN** **ELSE** check next surrounding cell
* **IF** the SOUTH cell is EXPLORED **THEN** **ELSE** check next surrounding cell
* **IF** the EAST cell is EXPLORED **THEN**  **ELSE** check next surrounding cell
* **IF** the WEST cell is EXPLORED **THEN** **ELSE** check next state
* **IF** the NORTH cell is UNKOWN **THEN** **ELSE** check next surrounding cell
* **IF** the SOUTH cell is UNKOWN **THEN** **ELSE** check next surrounding cell
* **IF** the EAST cell is UNKOWN **THEN**  **ELSE** check next surrounding cell
* **IF** the WEST cell is UNKOWN **THEN** **ELSE** check next surrounding cell

Could be the case that following the stated rules the robot gets stacked because none of the surrounding cells is available for visiting due to there is an OBSTACLE or is already VISITED. In that case, we need to move the robot to the closest available (EXPLORED) cell. To find that cell, we compute the Euclidean distance from the actual position to all the cells labeled as EXPLORED and choose the cell which has the lowest distance.

**Move:** Once that we know our next , there are two possibilities: In the first scenario, the is adjacent to the actual position, which means that is safe to move just by a small movement. In the second scenario, the is far away from the actual position, so, we need to move a large distance, according with the collected information, the robots doesn’t know for sure if the path is free of obstacles, therefore is necessary to activate an obstacle avoidance task that is described in the section XXX.