**PERCEPTION**

PROBLEM STATEMENT:

* TO CLEAR THE NOISE IN THE IMAGE.
* TO FIND THE OPTIMAL PATH FROM START POINT TO END POINT WITH EACH VERTEX OF THE GRID HAVING DIFFERENT COST.

FUNCTIONS AND ALOGORITHMS

* isValid()-checks the validity of the coordinates
* filter()-removes noise in the image.It reads the start and end point as coordinates and as the coordinate of end point is the length of side of each square,it traverses square by square and removes all the black points within each square.
* cost()-It returns the cost of traversing each pixel,i.e,from current pixel to next pixel as per the given costs.
* heuristics()-It calculates and returns the manhattan distance between the current node and start point.
* min\_prior()-It returns the co-ordinates of the smallest element in the array,passed by the calling function.
* bfs()-It calculates the path between the given points.Firstly,this function stores the respective cost of each pixel in the image,in the array cost\_so\_far by traversing the entire image and makes sure the calculated cost is minimum.Alongwith this another stores came\_from stores the coordinate of the parent pixel from which each pixel came.Now,after traversal we backtrack the path using the came\_from array uptil start point.
* dijkstra()-It calculates more optimal path between the given points.Firstly,this function stores the respective cost of each pixel traversed in the image,in the array 'cost\_so\_far' by traversing the entire image and makes sure the calculated cost is minimum.In another array priority[],we store the cost alongwith the heuristics of each pixel traversed.Also to make the traversal short and less time consuming,I made it traverse only the vertex of the grid as the cost of traversal of black points is zero.
* reconstruct\_path()-Using this function,I bactracked the path using the priority array,that is,by finding the minimum cost element in each 3X3 kernel around the current element and then making it the current element until we reach start point.
* main()-It finds the COM of the start and end points and calls the required function to execute the tasks.

**PROBLEMS:**

The following difficulties were encountered and sorted:

* I faced difficulty in implementing priority queue in my code which would have made the code more concise.As no effort bore fruit ,i switched to structured array and implemented my code.
* There were difficulties in running the code by traversing each pixel as it made it much time consuming, so I switched to Vertex Traversal Approach as the cost of black pixel was zero,which in turn reduced the time considerably.