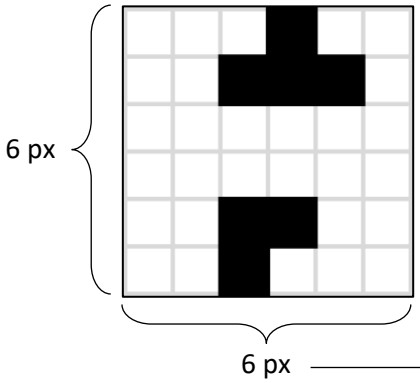
1.) Image (base_map.pgm) as starting point



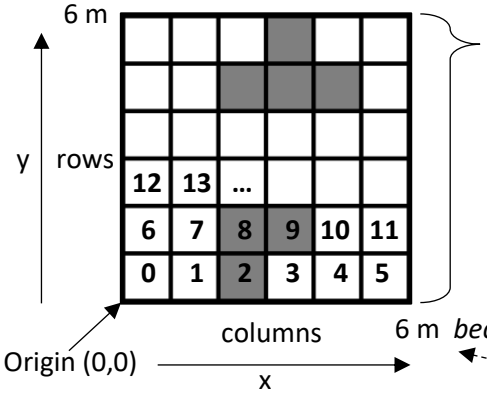
+ base_map.yaml with attributes like:

- image = base_map.pgm
- resolution = 1.0 (m/px)
- origin = [0.0, 0.0, 0.0]

(m/px=m/cell since px=cell for the map_server)

2.) Wrapper: ROS map_server

Reads yaml and pgm → creates array with an element for each pixel of base_map.pgm → stores array and metadata as OccupancyGrid → publishes it on "map" (base_map_topic)



Represented with OccupancyGrid (simplified):

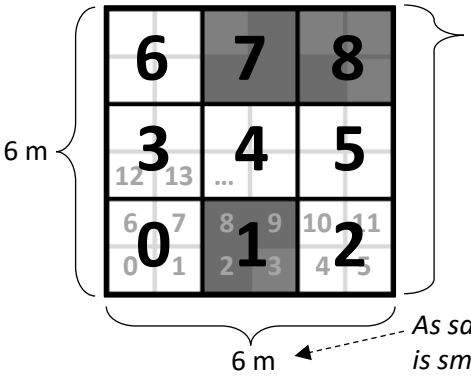
- height = 6 px
- width = 6 px
- resolution = 1.0 m/cell
- data = [0, 0, 100, 0, 0, 0,
  0, 0, 100, 100, 0, 0,
  0, 0, 0, 0, 0, 0, …] → **row-major order**

because: ***width · resolution = 6 px · 1 m/px = 6 m***

This real dimension will never change
(except if only a certain map section should be used)

3.) Wrapper: map_provider

Uses OccupancyGrid from "map" topic and does not change the general layout but might transform the array based on the resolution (we ignore for now the cropping of a map section), and publishes the transformed OccupancyGrid on "provided_map" (provided_map_topic) to e.g. simulator. Transformation example: User specified map_resolution = 2 m/cell in parameters.yaml:



Represented with OccupancyGrid (simplified):

- height = 3 px
- width = 3 px
- resolution = 2.0 m/cell
- data = [0, 100, 0,
  0, 0, 0,
  0, 100, 100] → **row-major order**

As said: the real map dimensions (in m) stay, but for example now the array is smaller (faster for path planning, etc. but loss of detailed map information)