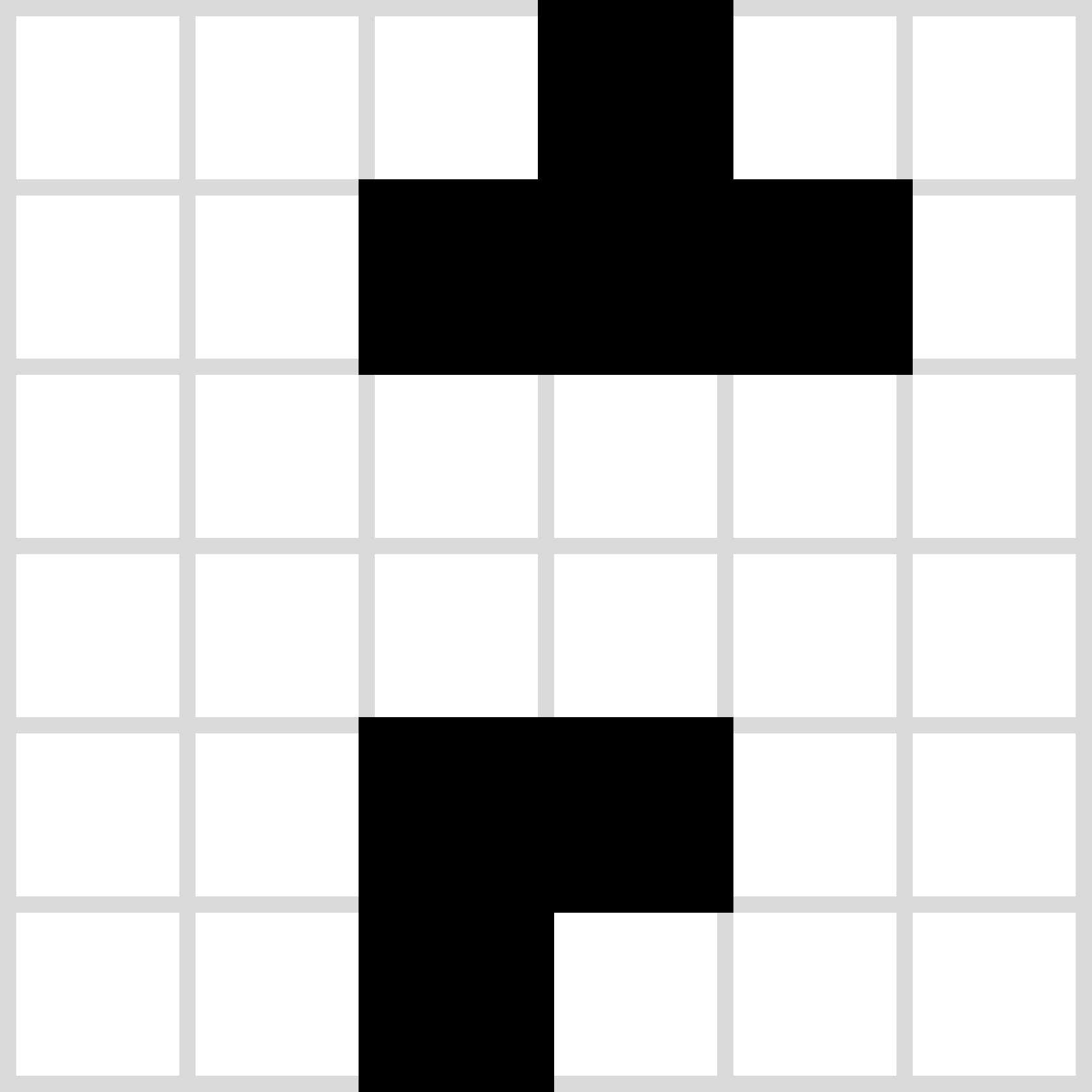
1. **Image (base\_map.pgm) as starting point**



6 px

6 px

+ base\_map.yaml with attributes like:

* image = base\_map.pgm
* resolution = 1.0 (m/px)

(m/px=m/cell since px=cell

for the map\_server)

* origin = [0.0, 0.0, 0.0]

1. **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)

6 m

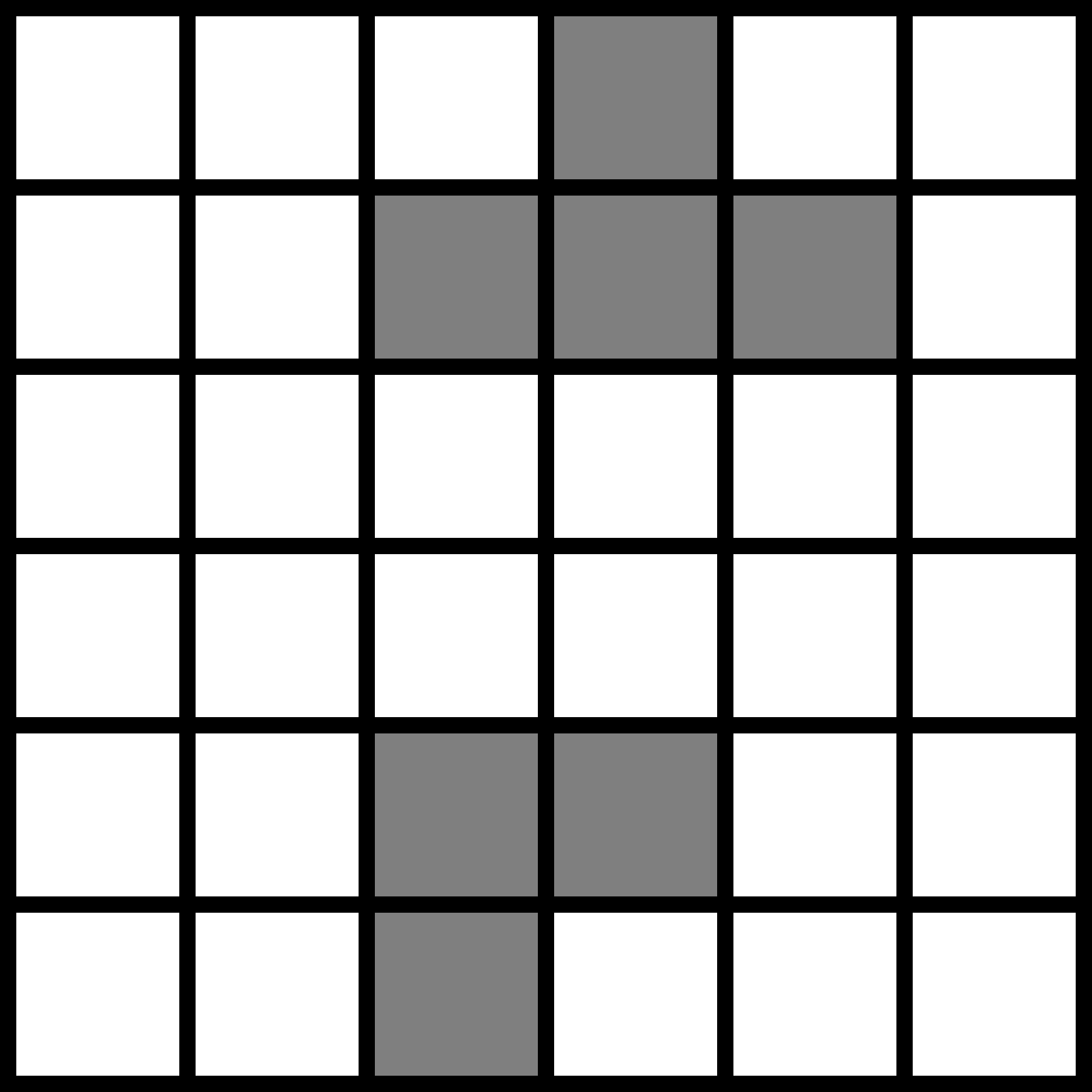
6 m  *because:* ***width · resolution = 6 px · 1 m/px = 6 m***

Origin (0,0)

y rows

x

columns



**0 1 2 3 4 5**

**6 7 8 9 10 11**

**12 13 …**

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**

*This real dimension will never change*

*(except if only a certain map section should be used)*

1. **Wrapper: map\_provider**

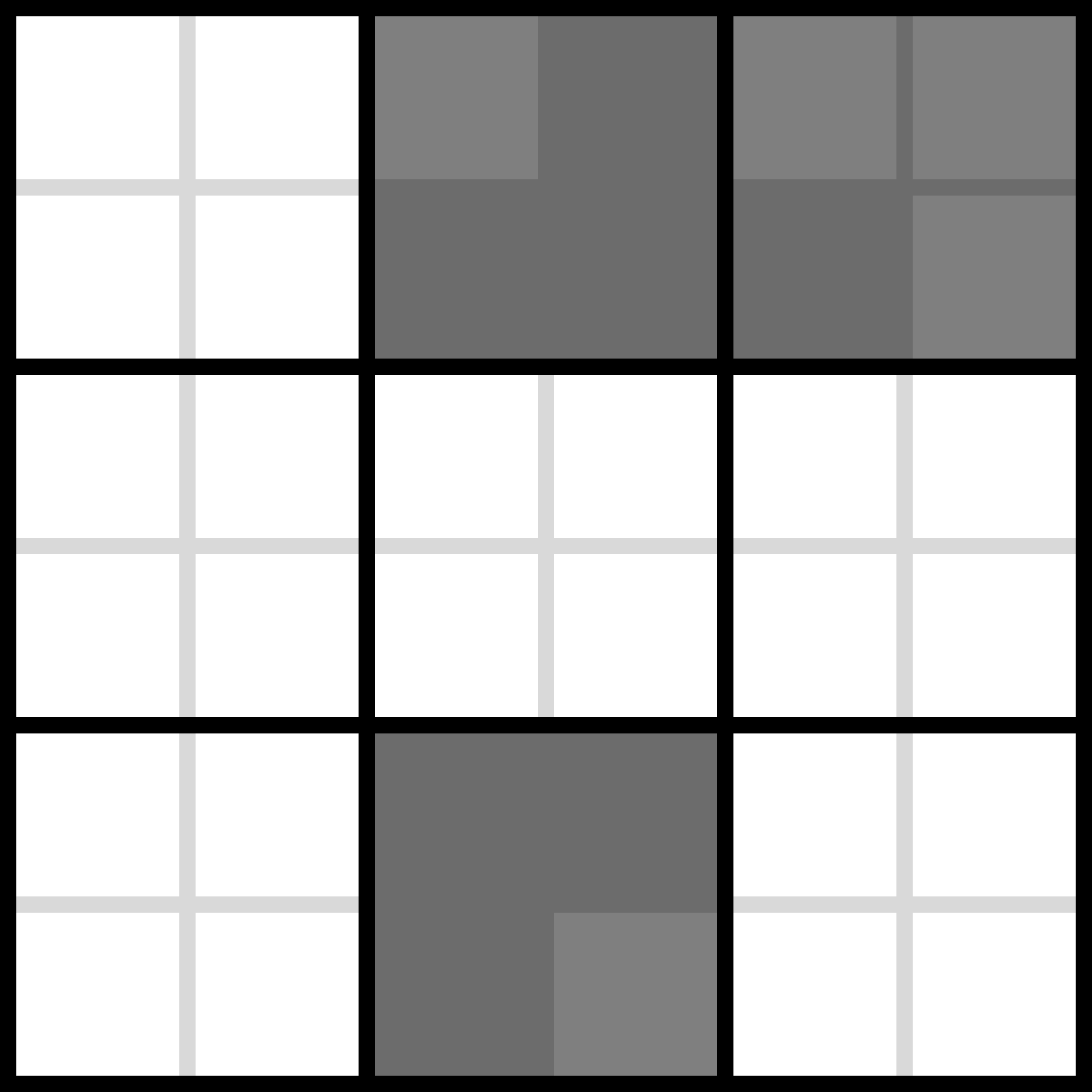
Uses OccupancyGrid from “map” topic and does not change the general layout but might transform the array based on the resolution (we ignore for now the cropping of a map section), and publishes the transformed OccupancyGrid on “provided\_map” (provided\_map\_topic) to e.g. simulator. Transformation example: 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**



**0 1 2 3 4 5**

**6 7 8 9 10 11**

**12 13 …**

**0 1 2**

**3 4 5**

**6 7 8**

6 m

*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)*

6 m