```
Algorithm motion_model_with_map(x_t, u_t, x_{t-1}, m):
    return p(x_t | u_t, x_{t-1}) · p(x_t | m)
```

```
Algorithm sample_motion_model_with_map(u_t, x_{t-1}, m):
    do
        x_t = sample_motion_model(u_t, x_{t-1})
        π = p(x_t | m)
    until π > 0
    return ⟨x_t, π⟩
```

Table 5.7 Algorithm for computing $p(x_t \mid u_t, x_{t-1}, m)$, which utilizes a map m of the environment. This algorithms bootstraps previous motion models (Tables 5.1, 5.3, 5.5, and 5.6) to models that take into account that robots cannot be placed in occupied space in the map m.

in the map; otherwise it assumes a constant value. By multiplying $p(x_t \mid m)$ and $p(x_t \mid u_t, x_{t-1})$, we obtain a distribution that assigns all probability mass to poses x_t consistent with the map, which otherwise has the same shape as $p(x_t \mid u_t, x_{t-1})$. As η can be computed by normalization, this approximation of a map-based motion model can be computed efficiently without any significant overhead compared to a map-free motion model.

Table 5.7 states the basic algorithms for computing and for sampling from the map-based motion model. Notice that the sampling algorithm returns a weighted sample, which includes an importance factor proportional to $p(x_t \mid m)$. Care has to be taken in the implementation of the sample version, to ensure termination of the inner loop. An example of the motion model is illustrated in Figure 5.11. The density in Figure 5.11a is $p(x_t \mid u_t, x_{t-1})$, computed according to the velocity motion model. Now suppose the map m possesses a long rectangular obstacle, as indicated in Figure 5.11b. The probability $p(x_t \mid m)$ is zero at all poses x_t where the robot would intersect the obstacle. Since our example robot is circular, this region is equivalent to the obstacle grown by a robot radius—this is equivalent to mapping the obstacle from *workspace* to the robot's *configuration space* or *pose space*. The resulting

CONFIGURATION SPACE