Simulation of Crowd Movement

Abdessamad Ed-dahmouni

March 2015

1 Model with no obstacles

We consider n agents in a room with one exit, agent i is modelled by a disk with center $M_i(x_i, y_i)$ and radius r_i moving by a discrete step p during time τ. $S(x_s, y_s)$ is the exit. The following two figures present the parameters used to describe the movement of agent i:

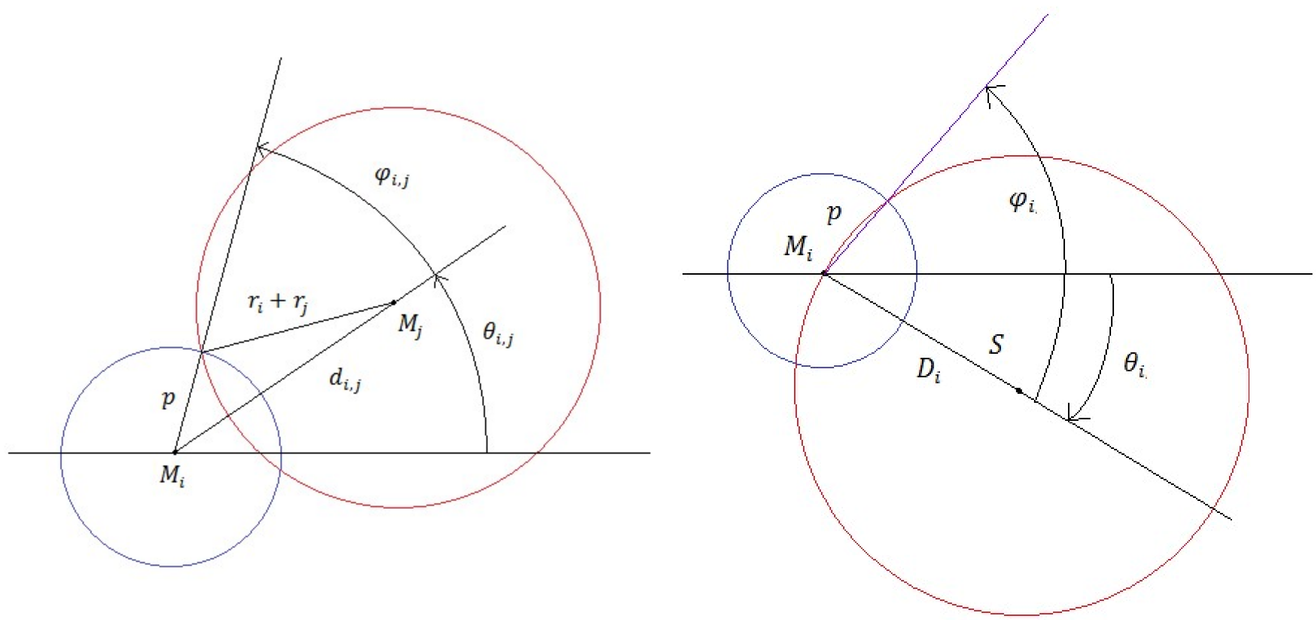

Figure 1: Parameters of the interaction between agents i and j

We find the following formulas:

$$\theta_i = \arctan(\frac{y_s - y_i}{x_s - x_i}) \quad \text{and} \quad \phi_i = \arccos(\frac{p}{2D_i})$$

And if $p + r_i + r_j > d_{i,j}$:

$$\theta_{i,j} = \arctan(\frac{y_j - y_i}{x_j - x_i}) \quad \text{and} \quad \phi_{i,j} = \arccos(\frac{p^2 + d_{i,j}^2 - (r_i + r_j)^2}{2d_{i,j}p})$$

If for agents i, and j we have $p + r_i + r_j > d_{i,j}$ we say that j blocks i.
We consider the case where i is blocked by multiple agents $K_i = \{k_1, \ldots, k_m\}$ (K_i could be empty), and we define:

$$I_{i,j} = [\theta_{i,j} - \phi_{i,j}, \theta_{i,j} + \phi_{i,j}]$$
$$I_i = [\theta_i - \phi_i, \theta_i + \phi_i]$$
$$S_i = (I_i \cap [-\frac{3\pi}{2}, \frac{3\pi}{2}]) \setminus \cup_{j \in K_i} I_{i,j}$$

If S_i is not empty, then, the optimal angle is:

$$\theta_m = \text{argmin}_{\alpha \in S_i} |\alpha - \theta_i|$$

I devised an algorithm for calculating this optimal value

2 Algorithm for Calculating the best direction

In this section, we write for calculating θ_m in the case of $K_i \neq \varnothing$. The other case is trivial, since we just have to choose $\theta_m = \theta_i$.

Algorithm 1 Optimal angle calculation

```
initialize u, l, upper, lower = θ_i and C = True:
while C do
    C = False
    for k in K_i do
        if θ_{i,k} − φ_{i,k} ≤ upper and u < θ_{i,k} + φ_{i,k} then
            u = θ_{i,k} + φ_{i,k}
            C = True
        end if
        if θ_{i,k} − φ_{i,k} < l and lower ≤ θ_{i,k} + φ_{i,k} then
            l = θ_{i,k} − φ_{i,k}
            C = True
        end if
    end for
    upper = u
    lower = l
end while
if |upper − θ_i| < |lower − θ_i| then
    return upper
else
    return lower
end if
```

I explored and implemented the Fast Marching Method in the case of obstacles but limited the interactions between agents to simple point collisions (the problem is solved on a regular grid, and it's hard to shift between the discrete space of positions and the continuous space of directions).
I used the **tkinter** Python library to create an interactive interface for the simulations.