Marco Tröster

October 27th, 2022

1 Repulsive Virtual Potential Fields

For avoiding obstacles, a repulsive force can be defined as virtual potential field. Obstacles are modeled as line segments:

$$obst = (P_{start}, P_{end}), P_{start} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, P_{end} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}, ped = \begin{pmatrix} x_3 \\ y_3 \end{pmatrix}$$

The formula for the potential can be defined as follows. Its' properties are that it becomes very large when a pedestrian is near the obstacle and diminishes when he's far away:

$$Pot(obst, ped) = \frac{1}{2 * dist(obst, ped)^2}$$

Furthermore, the repulsive force is just the inverse potential's derivative regarding the pedestrian's position.

$$Force(obst, ped) = -\frac{\partial Pot(obst, ped)}{\partial ped} = \frac{1}{dist(obst, ped)^3} \frac{\partial dist(obst, ped)}{\partial ped}$$

2 Modeling a Distance Function

After defining a potential field and a repulsive force, we now need to design a distance function modeling the distances between an obstacle and a pedestrian.

In our case, we use a formula for projecting the pedestrian's position towards a line segment which represents the obstacle. If an orthogonal projection from the pedestrian's position hits the line segment, we use the line/line intersection formula to retrieve the crossing point, then compute the euclidean distance between the pedestrian's position and the crossing point to retrieve the distance. The line segment is hit iff $0 \leq t \leq 1$.

$$\begin{pmatrix} x_4 \\ y_4 \end{pmatrix} = ped + |P_{start} P_{end}|^T = \begin{pmatrix} x_3 + v_x \\ y_3 + v_y \end{pmatrix}$$

$$t = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

$$P_{cross} = \begin{pmatrix} x_1 + t * (x_2 - x_1) \\ y_1 + t * (y_2 - y_1) \end{pmatrix}$$
$$dist(obst, ped) = euclid\_dist(P_{cross}, ped)$$

Otherwise, in case the projection doesn't hit the line segment, the pedestrian is off to the side and its' distance towards the obstacle can be retrieved by directly computing the euclidean distance between the closer line segment bound P_{start} or P_{end} and the pedestrian.

$$dist(obst, ped) = min(euclid\_dist(P_{start}, ped), euclid\_dist(P_{end}, ped))$$

For very rare cases, the obstacle is just a point instead of a line segment (i.e. $P_{start} = P_{end}$). Then, the distance can be trivially retrieved as the distance between the obstacle's position and the pedestrian's position. This case needs to be handled because the denominator of t becomes zero due to $x_1 - x_2 = y_1 - y_2 = 0$, causing math issues during computation; w.l.o.g. this case can be considered as if the projection doesn't hit the line segment.

$$dist(obst, ped) = euclid\_dist(P_{start}, ped)$$

3 Evaluation of the Repulsive Force Formula

Now that the distance is properly defined, the repulsive force can be evaluated. Recall that we need to figure out the partial derivative of our distance function by the pedestrian's position $ped = \begin{pmatrix} x_3 \\ y_3 \end{pmatrix}$.

$$Force(obst, ped) = -\frac{\partial Pot(obst, ped)}{\partial ped} = \frac{1}{dist(obst, ped)^3} \frac{\partial dist(obst, ped)}{\partial ped}$$

.

3.1 Projection doesn't hit the line segment (cases 2, 3)

For the second case where the projection doesn't hit the line segment, the distance's derivative is just the derivative of the euclidean distance function, denoted as $euclid\_dist(\begin{pmatrix} x_a \\ y_a \end{pmatrix}, \begin{pmatrix} x_b \\ y_b \end{pmatrix}) = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$.

$$\begin{aligned} \frac{\partial dist}{\partial x_a} &= \frac{\partial}{\partial x_a} \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \\ &= \frac{1}{2}((x_b - x_a)^2 + (y_b - y_a)^2)^{-\frac{1}{2}} \frac{\partial}{\partial x_a}((x_b - x_a)^2 + (y_b - y_a)^2) \\ &= \frac{1}{2dist}(\frac{\partial}{\partial x_a}(x_b - x_a)^2 + \frac{\partial}{\partial x_a}(y_b - y_a)^2) \\ &= \frac{1}{2dist} 2(x_b - x_a) \frac{\partial}{\partial x_a}(x_b - x_a) = \frac{x_a - x_b}{dist} \end{aligned} \tag{1}$$

$$
\begin{aligned}
\frac{\partial dist}{\partial y_a} &= \frac{\partial}{\partial y_a}\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \\
&= \frac{1}{2}((x_b - x_a)^2 + (y_b - y_a)^2)^{-\frac{1}{2}}(\frac{\partial}{\partial y_a}(x_b - x_a)^2 + (y_b - y_a)^2) \\
&= \frac{1}{2dist}(\frac{\partial}{\partial y_a}(x_b - x_a)^2 + \frac{\partial}{\partial y_a}(y_b - y_a)^2) \\
&= \frac{1}{2dist}2(y_b - y_a)\frac{\partial}{\partial y_a}(y_b - y_a) = \frac{y_a - y_b}{dist}
\end{aligned} \tag{2}
$$

In Python, this looks like the following snippet:

```
from math import dist

def der_euclid_dist(p1, p2):
    distance = dist(p1, p2)
    dx1_dist = (p1[0] - p2[0]) / distance
    dy1_dist = (p1[1] - p2[1]) / distance
    return dx1_dist, dy1_dist
```

3.2 Projection hits the line segment (case 1)

For the first case where the projection hits the line segment, the distance's derivative is a bit more complex as the crossing point P_{cross} depends on the pedestrian's position. Recall that P_{cross} is denoted as

$$
P_{cross} = \begin{pmatrix} x_1 + t * (x_2 - x_1) \\ y_1 + t * (y_2 - y_1) \end{pmatrix}, t = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}
$$

First, let's evaluate the terms $\frac{\partial t}{\partial x_3}$ and $\frac{\partial t}{\partial y_3}$ as they'll come in handy.

$$
\frac{\partial t}{\partial x_3} = \frac{\partial}{\partial x_3}\frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \quad = \frac{\partial}{\partial x_3}\frac{num}{den} \tag{3}
$$

$$
\frac{\partial t}{\partial y_3} = \frac{\partial}{\partial y_3}\frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \quad = \frac{\partial}{\partial y_3}\frac{num}{den} \tag{4}
$$

This is actually a lot to chew in one bite, so let's separate the evaluation into the derivatives of the numerator and denominator $\frac{\partial num}{\partial x_3}$, $\frac{\partial num}{\partial y_3}$, $\frac{\partial den}{\partial x_3}$, $\frac{\partial den}{\partial y_3}$, then combine the results with the quotient rule.

$$\begin{aligned}
\frac{\partial num}{\partial x_3} &= \frac{\partial}{\partial x_3}((x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)) \\
&= \frac{\partial}{\partial x_3}((x_1 - x_3)(y_3 - y_4)) - \frac{\partial}{\partial x_3}((y_1 - y_3)(x_3 - x_4)) \\
&= \frac{\partial}{\partial x_3}(x_1 - x_3)(y_3 - y_4) + \frac{\partial}{\partial x_3}(y_3 - y_4)(x_1 - x_3) - \frac{\partial}{\partial x_3}((y_1 - y_3)(x_3 - x_4)) \\
&= -1(y_3 - y_4) - \frac{\partial}{\partial x_3}((y_1 - y_3)(x_3 - x_4)) \\
&= -1(y_3 - y_4) - (\frac{\partial}{\partial x_3}(y_1 - y_3)(x_3 - x_4)\frac{\partial}{\partial x_3}(x_3 - x_4)(y_1 - y_3)) \\
&= -1(y_3 - y_4) - \frac{\partial}{\partial x_3}(x_3 - (x_3 + v_x))(y_1 - y_3) \\
&= y_4 - y_3
\end{aligned} \tag{5}$$

$$\begin{aligned}
\frac{\partial num}{\partial y_3} &= \frac{\partial}{\partial y_3}((x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)) \\
&= \frac{\partial}{\partial y_3}((x_1 - x_3)(y_3 - y_4)) - \frac{\partial}{\partial y_3}((y_1 - y_3)(x_3 - x_4)) \\
&= \frac{\partial}{\partial y_3}(x_1 - x_3)(y_3 - y_4) + \frac{\partial}{\partial y_3}(y_3 - y_4)(x_1 - x_3) - \frac{\partial}{\partial y_3}((y_1 - y_3)(x_3 - x_4)) \\
&= \frac{\partial}{\partial y_3}(y_3 - (y_3 + v_y))(x_1 - x_3) - \frac{\partial}{\partial y_3}((y_1 - y_3)(x_3 - x_4)) \\
&= -\frac{\partial}{\partial y_3}((y_1 - y_3)(x_3 - x_4)) \\
&= -(\frac{\partial}{\partial y_3}(y_1 - y_3)(x_3 - x_4) + \frac{\partial}{\partial y_3}(x_3 - x_4)(y_1 - y_3)) \\
&= -\frac{\partial}{\partial y_3}(y_1 - y_3)(x_3 - x_4) \\
&= x_3 - x_4
\end{aligned} \tag{6}$$

$$
\begin{aligned}
\frac{\partial den}{\partial x_3} &= \frac{\partial}{\partial x_3}((x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)) \\
&= \frac{\partial}{\partial x_3}((x_1 - x_2)(y_3 - y_4)) - \frac{\partial}{\partial x_3}((y_1 - y_2)(x_3 - x_4)) \\
&= -\frac{\partial}{\partial x_3}((y_1 - y_2)(x_3 - x_4)) \\
&= -(\frac{\partial}{\partial x_3}(y_1 - y_2)(x_3 - x_4) + \frac{\partial}{\partial x_3}(x_3 - x_4)(y_1 - y_2)) \\
&= -(\frac{\partial}{\partial x_3}(x_3 - (x_3 + v_x))(y_1 - y_2)) \\
&= 0
\end{aligned}
\tag{7}
$$

$$
\begin{aligned}
\frac{\partial den}{\partial y_3} &= \frac{\partial}{\partial y_3}((x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)) \\
&= \frac{\partial}{\partial y_3}((x_1 - x_2)(y_3 - y_4)) - \frac{\partial}{\partial y_3}((y_1 - y_2)(x_3 - x_4)) \\
&= \frac{\partial}{\partial y_3}((x_1 - x_2)(y_3 - y_4)) \\
&= \frac{\partial}{\partial y_3}(x_1 - x_2)(y_3 - y_4) + \frac{\partial}{\partial y_3}(y_3 - y_4)(x_1 - x_2) \\
&= \frac{\partial}{\partial y_3}(y_3 - (y_3 + v_y))(x_1 - x_2) \\
&= 0
\end{aligned}
\tag{8}
$$

For summary, $\frac{\partial num}{\partial x_3} = y_4 - y_3$, $\frac{\partial num}{\partial y_3} = x_3 - x_4$ and $\frac{\partial den}{\partial x_3} = \frac{\partial den}{\partial y_3} = 0$. Now we can come back to our original derivatives $\frac{\partial t}{\partial x_3}$ and $\frac{\partial t}{\partial y_3}$.

$$
\begin{aligned}
\frac{\partial t}{\partial x_3} = \frac{\partial}{\partial x_3}\frac{num}{den} &= \frac{\frac{\partial}{\partial x_3}num den - \frac{\partial}{\partial x_3}den num}{den^2} \quad = \frac{\frac{\partial}{\partial x_3}num den}{den^2} \\
&= \frac{(y_4 - y_3)den}{den^2} = \frac{y_4 - y_3}{den}
\end{aligned}
\tag{9}
$$

$$
\begin{aligned}
\frac{\partial t}{\partial y_3} = \frac{\partial}{\partial y_3}\frac{num}{den} &= \frac{\frac{\partial}{\partial y_3}num den - \frac{\partial}{\partial y_3}den num}{den^2} \quad = \frac{\frac{\partial}{\partial y_3}num den}{den^2} \\
&= \frac{(x_3 - x_4)den}{den^2} = \frac{x_3 - x_4}{den}
\end{aligned}
\tag{10}
$$

Next, we need to evaluate the equations yielding the crossing point P_{cross}. In particular, we need to evaluate $\frac{\partial cross_x}{\partial x_3}$, $\frac{\partial cross_y}{\partial x_3}$, $\frac{\partial cross_x}{\partial y_3}$, $\frac{\partial cross_y}{\partial y_3}$.

Recall that our crossing point is still denoted as

$$P_{cross} = \begin{pmatrix} x_1 + t * (x_2 - x_1) \\ y_1 + t * (y_2 - y_1) \end{pmatrix}$$

$$\begin{aligned} \frac{\partial cross_x}{\partial x_3} &= \frac{\partial}{\partial x_3} x_1 + t(x_2 - x_1) \\ &= \frac{\partial}{\partial x_3} x_1 + \frac{\partial}{\partial x_3}(t(x_2 - x_1)) \\ &= \frac{\partial}{\partial x_3} t(x_2 - x_1) + \frac{\partial}{\partial x_3}(x_2 - x_1)t \\ &= \frac{y_4 - y_3}{den}(x_2 - x_1) \end{aligned} \tag{11}$$

Similarly to the previous proof sketch, $\frac{\partial cross_y}{\partial x_3} = \frac{y_4 - y_3}{den}(y_2 - y_1)$, $\frac{\partial cross_x}{\partial y_3} = \frac{x_3 - x_4}{den}(x_2 - x_1)$ and $\frac{\partial cross_y}{\partial y_3} = \frac{x_3 - x_4}{den}(y_2 - y_1)$.

Last, we evaluate the full derivative of the euclidean distance.

$$\begin{aligned} \frac{\partial dist}{\partial x_3} &= \frac{\partial}{\partial x_3} \sqrt{(cross_x - x_3)^2 + (cross_y - y_3)^2} \\ &= \frac{1}{2dist} \frac{\partial}{\partial x_3}((cross_x - x_3)^2 + (cross_y - y_3)^2) \\ &= \frac{1}{2dist}(\frac{\partial}{\partial x_3}(cross_x - x_3)^2 + \frac{\partial}{\partial x_3}(cross_y - y_3)^2) \\ &= \frac{1}{2dist}(2(cross_x - x_3)\frac{\partial}{\partial x_3}(cross_x - x_3) + \frac{\partial}{\partial x_3}(cross_y - y_3)^2) \\ &= \frac{1}{2dist}(2(cross_x - x_3)(\frac{\partial}{\partial x_3} cross_x - \frac{\partial}{\partial x_3} x_3) + \frac{\partial}{\partial x_3}(cross_y - y_3)^2) \\ &= \frac{1}{2dist}(2(cross_x - x_3)(\frac{\partial}{\partial x_3} cross_x - 1) + \frac{\partial}{\partial x_3}(cross_y - y_3)^2) \\ &= \frac{1}{2dist}(2(cross_x - x_3)(\frac{\partial}{\partial x_3} cross_x - 1) + 2(cross_y - y_3)\frac{\partial}{\partial x_3}(cross_y - y_3)) \\ &= \frac{1}{2dist}(2(cross_x - x_3)(\frac{\partial}{\partial x_3} cross_x - 1) + 2(cross_y - y_3)(\frac{\partial}{\partial x_3} cross_y - \frac{\partial}{\partial x_3} y_3)) \\ &= \frac{1}{2dist}(2(cross_x - x_3)(\frac{\partial}{\partial x_3} cross_x - 1) + 2(cross_y - y_3)\frac{\partial}{\partial x_3} cross_y) \\ &= \frac{1}{dist}((cross_x - x_3)(\frac{\partial}{\partial x_3} cross_x - 1) + (cross_y - y_3)\frac{\partial}{\partial x_3} cross_y) \end{aligned} \tag{12}$$

Similarly, $\frac{\partial dist}{\partial y_3}$ can be retrieved as

$$\frac{\partial dist}{\partial y_3} = \frac{1}{dist}((cross_x - x_3)(\frac{\partial}{\partial y_3} cross_x) + (cross_y - y_3)(\frac{\partial}{\partial y_3} cross_y - 1))$$

.

Following snippet outlines a possible implementation in Python:

```python
def obstacle_force(obstacle, ortho_vec, ped_pos):
    x1, y1, x2, y2 = obstacle
    x3, y3 = ped_pos
    v_x, v_y = ortho_vec
    x4, y4 = x3 + v_x, y3 + v_y

    # handle edge case where the obstacle is just a point
    if (x1, y1) == (x2, y2):
        obst_dist = euclid_dist(ped_pos[0], ped_pos[1], x1, y1)
        dx_obst_dist, dy_obst_dist = der_euclid_dist(
            ped_pos, (x1, y1), obst_dist)
        return potential_field_force(
            obst_dist, dx_obst_dist, dy_obst_dist)

    num = (x1 - x3) * (y3 - y4) - (y1 - y3) * (x3 - x4)
    den = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4)
    t = num / den
    ortho_hit = 0 <= t <= 1

    # orthogonal vector doesn’t hit within segment bounds
    if not ortho_hit:
        d1 = euclid_dist(ped_pos[0], ped_pos[1], x1, y1)
        d2 = euclid_dist(ped_pos[0], ped_pos[1], x2, y2)
        obst_dist = min(d1, d2)
        closer_obst_bound = (x1, y1) if d1 < d2 else (x2, y2)
        dx_obst_dist, dy_obst_dist = der_euclid_dist(
            ped_pos, closer_obst_bound, obst_dist)
        return potential_field_force(
            obst_dist, dx_obst_dist, dy_obst_dist)

    # orthogonal vector hits within segment bounds
    cross_x, cross_y = x1 + t * (x2 - x1), y1 + t * (y2 - y1)
    obst_dist = euclid_dist(x3, y3, cross_x, cross_y)
    dx3_cross_x = (y4 - y3) / den * (x2 - x1)
    dx3_cross_y = (y4 - y3) / den * (y2 - y1)
    dy3_cross_x = (x3 - x4) / den * (x2 - x1)
    dy3_cross_y = (x3 - x4) / den * (y2 - y1)
    dx_obst_dist = ((cross_x - ped_pos[0]) * (dx3_cross_x - 1) \
        + (cross_y - ped_pos[1]) * dx3_cross_y) / obst_dist
    dy_obst_dist = ((cross_x - ped_pos[0]) * dy3_cross_x \
        + (cross_y - ped_pos[1]) * (dy3_cross_y - 1)) / obst_dist
    return potential_field_force(
        obst_dist, dx_obst_dist, dy_obst_dist)
```

Finally, to compute the accumulated obstacle forces for each pedestrian, the forces for each obstacle need to be summed up by x / y components.

```python
for i in range(num_peds):
    ped_pos = ped_positions[i]
    for j in range(num_obstacles):
        force_x, force_y = obstacle_force(
            obstacle_segments[j], ortho_vecs[j], ped_pos)
        out_forces[i, 0] += force_x
        out_forces[i, 1] += force_y
```