”به نام یزدان پاک“

به صورت خلاصه:

در فاز اول ورودی ها را به صورت فازی در آورید.(fuzzification)

در فاز دوم خروجی را به صورت فازی به کمک قواعد در آورید.(inference)

در فاز سوم خروجی فازی شده را به صورت مقدار مطلق در آورید.(defuzzification)

در ابتدای کار ما معادله خط نمودار های pv, force ,pa و cv (برای قسمت امتیازی) را با استفاده از دو نقطه که در simple.fcl است محاسبه میکنیم.

سپس قوانین که در simple.fcl است را پیاده سازی میکنیم به طوری که and معادل min و or برابر max است.

```
'''
    RULE'S
'''
def left_fast_rules(self, pa, pv, cv):
    lf1 = min(self.up_more_left(pa), self.cw_slow(pv))
    lf2 = min(self.up_more_left(pa), self.ccw_slow(pv))
    lf3 = min(self.up_more_left(pa), self.ccw_fast(pv))
    lf4 = min(self.down_more_left(pa), self.cw_slow(pv))
    lf5 = min(self.down_left(pa), self.cw_slow(pv))
    lf6 = min(self.down_left(pa), self.ccw_slow(pv))
    lf7 = min(self.up_left(pa), self.ccw_slow(pv))
    lf8 = min(self.up_left(pa), self.stop_pv(pv))
    lf9 = min(self.up_right(pa), self.ccw_fast(pv))
    lf10 = min(self.up_left(pa), self.ccw_fast(pv))
    lf11 = min(self.up(pa), self.ccw_fast(pv))

    return max(lf1, lf2, lf3, lf4, lf5, lf6, lf7, lf8, lf9,
lf10, lf11)
```

```
def left_slow_rules(self, pa, pv, cv):
    ls1 = min(self.up_more_right(pa), self.ccw_fast(pv))
    ls2 = min(self.down_left(pa), self.ccw_fast(pv))
    ls3 = min(self.up_left(pa), self.cw_slow(pv))
    ls4 = min(self.up(pa), self.ccw_slow(pv))

    return max(ls1, ls2, ls3, ls4)

def stop_rules(self, pa, pv, cv):
    s1 = min(self.down_more_right(pa), self.cw_slow(pv))
    s2 = min(self.down_more_left(pa), self.ccw_slow(pv))
    s3 = min(self.down_more_right(pa), self.ccw_fast(pv))
    s4 = min(self.down_more_right(pa), self.cw_fast(pv))
    s5 = min(self.down_more_left(pa), self.cw_fast(pv))
    s6 = min(self.down_more_left(pa), self.ccw_fast(pv))
    s7 = min(self.down_right(pa), self.ccw_fast(pv))
    s8 = min(self.down_left(pa), self.cw_fast(pv))
    s9 = min(self.down(pa), self.cw_fast(pv))
    s10 = min(self.down(pa), self.ccw_fast(pv))
    s11 = min(self.up(pa), self.stop_pv(pv))
    s12 = max(min(self.up(pa), self.stop_pv(pv)),
min(self.up_right(pa), self.ccw_slow(pv)),
             min(self.up_left(pa), self.cw_slow(pv)))

    return max(s1, s2, s3, s4, s5, s6, s7, s8, s9, s10,
s11, s12)

def right_slow_rules(self, pa, pv, cv):
    rs1 = min(self.up_more_left(pa), self.cw_fast(pv))
    rs2 = min(self.down_right(pa), self.cw_fast(pv))
    rs3 = min(self.up_right(pa), self.ccw_slow(pv))
    rs4 = min(self.up(pa), self.cw_slow(pv))

    return max(rs1, rs2, rs3, rs4)

def right_fast_rules(self, pa, pv, cv):
    rf1 = min(self.up_more_right(pa), self.ccw_slow(pv))
    rf2 = min(self.up_more_right(pa), self.cw_slow(pv))
    rf3 = min(self.up_more_right(pa), self.cw_fast(pv))
    rf4 = min(self.down_more_right(pa), self.ccw_slow(pv))
    rf5 = min(self.down_right(pa), self.ccw_slow(pv))
    rf6 = min(self.down_right(pa), self.cw_slow(pv))
    rf7 = min(self.up_right(pa), self.cw_slow(pv))
    rf8 = min(self.up_right(pa), self.stop_pv(pv))
```

```
    rf9 = min(self.up_right(pa), self.cw_fast(pv))
    rf10 = min(self.up_left(pa), self.cw_fast(pv))
    rf11 = min(self.down(pa), self.stop_pv(pv))
    rf12 = min(self.up(pa), self.cw_fast(pv))

    return max(rf1, rf2, rf3, rf4, rf5, rf6, rf7, rf8, rf9,
rf10, rf11, rf12)
```

(خطوط بالا قوانین است که در دستورکار به آن اشاره شده)

در ادامه خطوطی است که ما به کد اضافه کرده ایم که با استفاده از cv کار میکنند:

```
lf12 = min(self.up_more_left(pa), self.right_fast_cv(cv))
```

```
ls5 = min(self.up_left(pa), self.right_slow_cv(cv))
ls6 = min(self.up_more_left(pa), self.right_fast_cv(cv))
```

```
s13 = min(self.up_more_right(pa), self.right_fast_cv(cv))
s14 = min(self.up_more_left(pa), self.left_fast_cv(cv))
s15 = min(self.up_more_right(pa), self.right_slow_cv(cv))
s16 = min(self.up_more_left(pa), self.left_slow_cv(cv))
```

```
rs5 = min(self.up_right(pa), self.left_slow_cv(cv))
```

```
rf13 = min(self.up_more_right(pa), self.left_fast_cv(cv))
```

در ادامه با استفاده از قوانین و مقایسه آنها با pa, pv, pc داده شده در input و پیدا کردن max آنها و بدست آوردن مجموع force ها و مجموع force*i که موقعیت آونگ است و تقسیم آنها بر یکدیگر میتوان نیرو وارد شده را بدست آورد.

```
def calculate(self, input):
    pa, pv, cv = input['pa'], input['pv'], input['cv']
    lf = self.left_fast_rules(pa, pv, cv)
    ls = self.left_slow_rules(pa, pv, cv)
    s = self.stop_rules(pa, pv, cv)
    rs = self.right_slow_rules(pa, pv, cv)
    rf = self.right_fast_rules(pa, pv, cv)
    i = -100
    sum1 = 0
    sum2 = 0
    while i <= 100:
        lf_force = self.left_fast(i)
        ls_force = self.left_slow(i)
```

```
        s_force = self.stop(i)
        rs_force = self.right_slow(i)
        rf_force = self.right_fast(i)

        lf_total = min(lf, lf_force)
        ls_total = min(ls, ls_force)
        s_total = min(s, s_force)
        rs_total = min(rs, rs_force)
        rf_total = min(rf, rf_force)

        force_total = max(lf_total, ls_total, s_total, rs_total,
rf_total)
        sum1 += force_total
        sum2 += force_total * i
        i += 0.5
    if sum1==0:
        return 0
    return sum2 / sum1
```

و در پایان کار آونگ ما به دلیل اعمال نیروی مناسب در جهت عمود بر گاری حرکت میکند.