گزارشکار پروژه فازی هوش محاسباتی

آوا اسماعیلی-۹۹۳۱۰۶۴

فاز اول

بخش اول (فازی سازی):

برای فازی سازی ورودی‌ها از تابع‌های تعلق استفاده میکنیم. به این صورت که معادله خط حساب کرده و تابع‌ها را پیاده‌سازی می‌کنیم. در هر یک از تابع‌های مذکور یک x به عنوان وروی گرفته می‌شود و با توجه به بازه تعریف شده به ما یک مقدار تعلق می‌دهد.

تابع‌های تعلق فاصله از راست:

```
def close_R(self, x):...

def moderate_R(self, x):...

def far_R(self, x):...
```

تابع‌های تعلق فاصله از چپ:

```
def close_L(self, x):...

def moderate_L(self, x):...

def far_L(self, x):...
```

تابع‌های تعلق چرخش فرمان:

```
def high_right(self, x):...

def low_right(self, x):...

def nothing(self, x):...

def high_left(self, x):...

def low_left(self, x):...
```

بخش دوم (نتیجه‌گیری):

در این مرحله مقادیر فازی به دست آورده را با منطق فازی بررسی می‌کنیم. به این منظور از قوانین نوشته شده در فایل rules.txt استفاده می‌کنیم.

```
IF (d_L IS close_L ) AND (d_R IS moderate_R)  THEN  Rotate IS low_right
IF (d_L IS close_L ) AND (d_R IS far_R)  THEN  Rotate IS high_right
IF (d_L IS moderate_L ) AND (d_R IS close_R)  THEN  Rotate IS low_left
IF (d_L IS far_L ) AND (d_R IS close_R)  THEN  Rotate IS high_left
IF (d_L IS moderate_L ) AND (d_R IS moderate_R)  THEN  Rotate IS nothing
```

با استفاده از مینیمم گرفتن از هر بخش قوانین بالا را اجرا می‌کنیم:

```
high_right = min(self.close_L(left_dist), self.far_R(right_dist))
low_right = min(self.close_L(left_dist), self.moderate_R(right_dist))
nothing = min(self.moderate_L(left_dist), self.moderate_R(right_dist))
high_left = min(self.far_L(left_dist), self.close_R(right_dist))
low_left = min(self.moderate_L(left_dist), self.close_R(right_dist))
```

سوال امتیازی:
می‌توانید از عملگرهای aggregation برای ترکیب مقادیر تعلق در چندین قانون فعال استفاده کنید. یکی از روش‌های معمول برای تجمیع این مقادیر، استفاده از عملگر ماکزیمم است. به این صورت که مقدار تعلق نهایی برابر با بیشترین مقدار تعلق در میان قوانین فعال قرار می‌گیرد. به طور کلی، با استفاده از عملگرهای تجمیع فازی، می‌توانید مقادیر تعلق مختلف را ترکیب کرده و به یک مقدار تعلق نهایی برسید. عملگرهای دیگری مانند میانگین و مرکزیت نیز می‌توانند در برخی موارد استفاده قرار بگیرند، اما عملگر ماکزیمم به صورت عمومی برای تجمیع مقادیر تعلق در قوانین فعال پیشنهاد می‌شود.
لازم به ذکر است که روش ترکیب مقادیر تعلق بستگی به نوع قوانین و نوع مسئله دارد. برای حالت‌های خاصتر ممکن است روش‌های دیگری مانند انتگرال‌گیری فازی (fuzzy integral) یا عملگرهای تجمیع دیگر مورد استفاده قرار بگیرند.

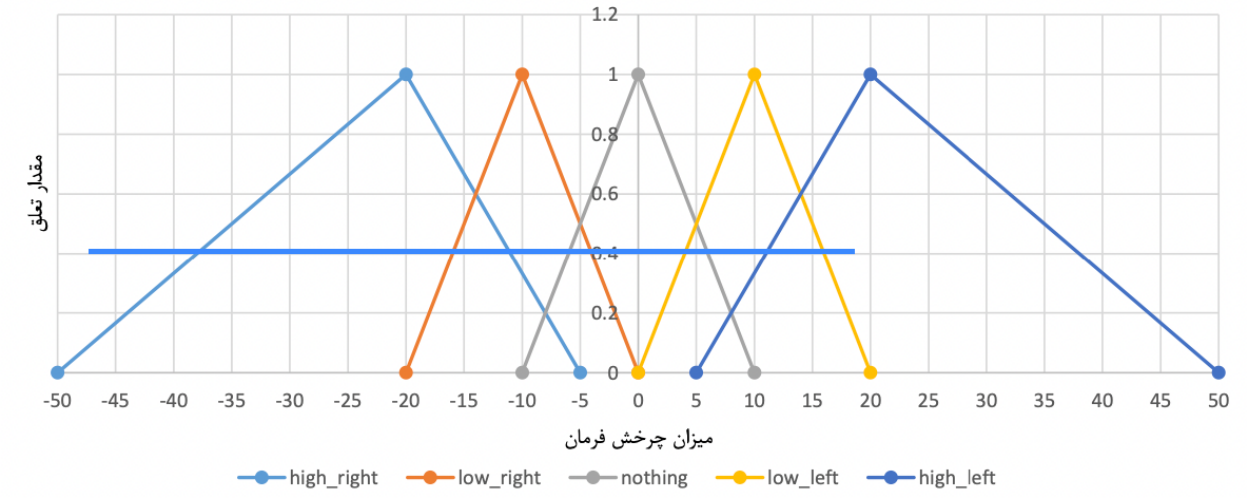
بخش سوم (فازی زدایی):

حال نتایجی را که در بخش قبل بدست آوردیم را به صورت عدد مطلقی در می‌آوریم که درواقع درجه چرخش فرمان است.

اگر مقادیر قسمت قبل به صورت زیر باشد:

high_right = 0.4, low_right = 0, nothing = 0, high_left=0, low_left = 0

و نمودار را به صورت زیر در نظر بگیریم:



و در نهایت فقط نمودار high_right تا خطی که رسم کردیم باقی می‌ماند.

این مرحله به صورت زیر پیاده‌سازی می‌شود:

```
def max_func(x):
    return max(min(low_right, self.low_right(x)),
               min(high_right, self.high_right(x)),
               min(low_left, self.low_left(x)),
               min(high_left, self.high_left(x)),
               min(nothing, self.nothing(x)))
```

برای پیاده‌سازی کد انتگرال‌گیری و محاسبه مرکز جرم از شبه‌کد دستورکار استفاده می‌کنیم:

```
s = 0.0
m = 0.0
X = self.linspace(-50, 50, 1000)
delta = X[1] - X[0]
for i in X:
    U = max_func(i)
    s += U * i * delta
    m += U * delta
center = 0.0
if m != 0:
    center = 1.0 * float(s) / float(m)
```

فاز دوم

در این فاز هدف این است که از سنسور سومی نیز استفاده کنیم که هدف آن تنظیم سرعت است.

بخش اول (فازی سازی):

توضیحات مانند فاز قبل است فقط تابع‌های تعلق تعریف شده این فاز باید پیاده‌سازی شود که شامل فاصله از جلو و سرعت است:

```
def close_c(self, x):...

def moderate_c(self, x):...

def far_c(self, x):...

def low_speed(self, x):...

def medium_speed(self, x):...

def high_speed(self, x):...
```

بخش دوم (نتیجه‌گیری):

در این مرحله مقادیر فازی به دست آورده را با منطق فازی بررسی می‌کنیم. به این منظور از قوانین نوشته شده در فایل additional_rules.txt استفاده می‌کنیم.

```
IF (center_dist IS close )  THEN  gas IS low
IF (center_dist IS moderate )  THEN  gas IS medium
IF (center_dist IS far )  THEN  gas IS high
```

به این صورت قوانین بالا را اجرا می‌کنیم:

```
low = self.close_c(center_dist)
medium = self.moderate_c(center_dist)
high = self.far_c(center_dist)
```

بخش سوم (فازی زدایی):

با همان توضیحات بالا بین نمودارها ماکس گرفته و انتگرال گیری انجام می‌دهیم:

```
def max_func(x):
    return max(min(low, self.low_speed(x)),
               min(high, self.high_speed(x)),
               min(medium, self.medium_speed(x)))
```

انتگرال گیری ما به صورت فاز یک است و فقط بازه ما از ۰ تا +۹۰ تغییر می‌کند:

```
s = 0.0
m = 0.0
X = self.linspace(0, 90, 1000)
delta = X[1] - X[0]
for i in X:
    U = max_func(i)
    s += U * i * delta
    m += U * delta
center = 0.0
if m != 0:
    center = 1.0 * (float(s)) / (float(m))
```