Gestion des panneaux stop

Dans le dossier « Parcour_urbain\AMI_With_GPS », vous trouverez le diagramme RT-MAPS « DiagrammeAvecGPSrunDisk.rtd » ainsi que « DiagrammeAvecGPSrunDiskSurAmi.rtd »

Ce dernier correspond au premier diagramme mais avec l'ensemble des connexions avec la voiture.

Le principe de la gestion des panneaux :

Nous utilisons la caméra realsense située devant la voiture pour détecter les panneaux à l'aide d'une cascade de Haar avec un classifieur xml. Une fois détecter, nous entourons le panneau avec un carré vert. Nous mesurons la largeur de ce carré afin de pouvoir calculer la distance à laquelle se trouve le panneau. Avec cette distance, nous la comparons à la distance d'arrêt. S'il elle est plus grande alors on pile, sinon on applique un certain niveau de frein en fonction de la vitesse, celui-ci va de 0 à 2 (valeur envoyée à la voiture pour le frein)

Plus précisément nous avons tracé une courbe modélisant les distances du panneau en fonction de sa largeur. Le code « realsense.py » présent dans « Parcours_urbain\Vscode » permet d'afficher les largeurs du panneau en pixel lorsqu'il en détecte un. Attention il faut brancher une realsense pour que le code fonctionne. Vous devrez donc réaliser de nouveaux la mesure pour le panneau présent à l'utac s'il ne mesure pas 42 cm de largeur.

Une fois les valeurs mesurées, il faut les remplacer dans le code python « realsenseVitesseStop.py » présent dans \Parcour_urbain\AMI_With_GPS\python_script :

```
61          x_data = np.array([120,110,100,86,68, 56, 51, 45, 38, 33,30,28,27,26,25])
62          y_data = np.array([200,220,250,300,400, 500, 600, 700, 800, 900,1000,1100,1200,1300,1400])
```

X_data correspond aux largeurs du panneaux en pixel et y_data à la distance à laquelle le panneau se situe. Par exemple à 8 mètres (800cm le panneau mesure 38 pixels)

realsenseVitesseStop.py :

```
108         self.histoStop = [0,0,0,0,0,0,0,0,0,0]
109         self.compteur = 0
```

Nous réalisons ici un tableau de zéro ainsi qu'un compteur. Le but est de savoir lorsque la voiture ne voit plus de façon continue le panneau, si dans les 10 frames précédentes le panneau a été détecté plus de 7 fois alors nous n'avons pas de faux positifs. Il serait possible de le vérifier avec une mesure de profondeur via la realsense.

```
if istop == 1 :
    self.histoStop[self.compteur]=1
    self.compteur = (self.compteur + 1)%10
    self.outputs["isStop"].write(1)
else :
    self.histoStop[self.compteur]=0
    self.compteur = (self.compteur + 1)%10
    self.outputs["isStop"].write(0)
if (sum(self.histoStop)/len(self.histoStop))>=0.7:
    self.outputs["averageStop"].write(1)
else :
    self.outputs["averageStop"].write(0)
```

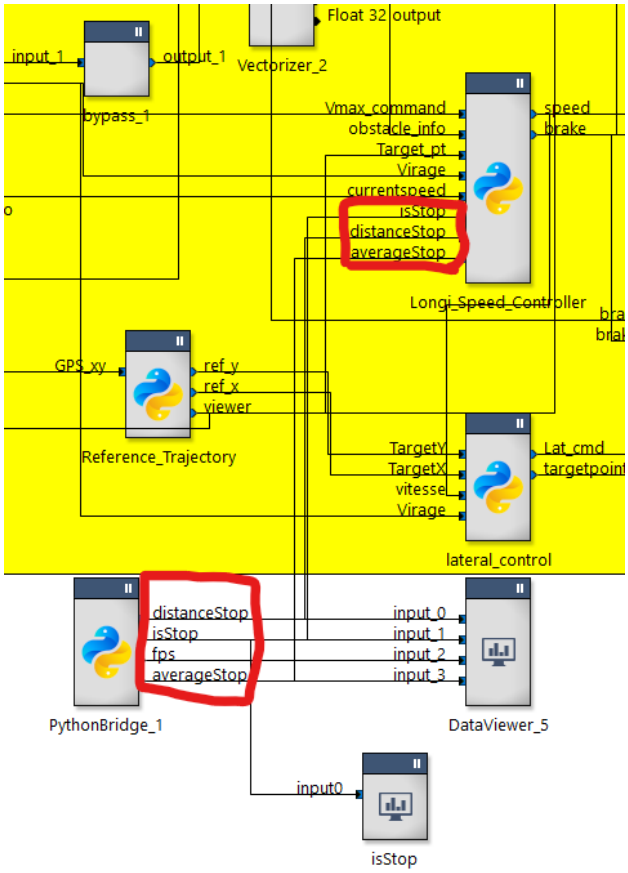
```
istop=0
for (x, y, w, h) in panneauxStop:
    detected_signsStop.append({'x': x, 'y': y, 'width': w, 'height': h})
for sign in detected_signsStop:
    x, y, w, h = sign['x'], sign['y'], sign['width'], sign['height']
    if(w<120) :
        distance = (lambda x, a, b, c, d, e: a*x**4 + b*x**3 + c*x**2 + d*x + e)(w, *self.params)
    else :
        distance = 200
    cv2.rectangle(color_image2, (x, y), (x+w, y+h), (0, 255, 0), 2)
    # self.outputs["xStop"].write(x)
    # self.outputs["yStop"].write(y)
    # self.outputs["wStop"].write(w)
    # self.outputs["hStop"].write(h)
    self.outputs["distanceStop"].write(distance)
    istop = 1
if istop == 1 :
    self.histoStop[self.compteur]=1
    self.compteur = (self.compteur + 1)%10
    self.outputs["isStop"].write(1)
else :
    self.histoStop[self.compteur]=0
    self.compteur = (self.compteur + 1)%10
    self.outputs["isStop"].write(0)
if (sum(self.histoStop)/len(self.histoStop))>=0.7:
    self.outputs["averageStop"].write(1)
else :
    self.outputs["averageStop"].write(0)
end_time = cv2.getTickCount()  # Record the end time
elapsed_time = (end_time - start_time) / cv2.getTickFrequency()  # Calculate elapsed time in seconds
fps = 1.0 / elapsed_time  # Calculate frames per second
self.outputs["fps"].write(fps)
self.output_video.write(color_image)
```

Dans ce code, si la largeur du carré entourant le panneau détecté est inférieur à 120, nous appliquons la courbe de distance avec une fonction polynomiale de degré 4. Sinon nous estimons que le distance est de 200cm (correspondance avec notre courbe expliqué auparavant). Si un panneau est détecté (variable isStop à 1), nous remplissons le tableau histoStop avec une valeur de 1 (cela pour chaque frame) puis non augmentons le compteur de 1 le tout modulo 10 afin de vérifier la présence du stop tous les 10 frames. Si la moyenne des valeurs dans ce tableau est supérieure à 0.7, on estime qu'il y a réellement un panneau devant le véhicule. Cette méthode permet de savoir s'il y a bien un stop même quand il n'est plus détecté par la voiture (trop proche) mais également d'éviter les arrêts avec des faux positifs.

Longi_Speed_Trash.py :

```
100         if index ==5:
101             if self.stopEnable == 1:
102                 self.isStop = self.inputs["isStop"].ioelt.data
103                 self.averageStop = self.inputs["averageStop"].ioelt.data
```

Nous avons ajouté le cas du stop dans l'index 5 qui récupère la valeur des variables isStop et averageStop envoyées par le bloc précédent (realsenseVitesseStop.py) (voir photo ci-dessous).


Float 32 output
input_1
output_1
Vectorizer_2
bypass_1
Vmax_command
obstacle_info
Target_pt
Virage
currentspeed
isStop
distanceStop
averageStop
speed
brake
Longi_Speed_Controller
GPS_xy
ref_y
ref_x
viewer
Reference_Trajectory
TargetY
TargetX
vitesse
Virage
Lat_cmd
lateral_control
distanceStop
isStop
fps
averageStop
input_0
input_1
input_2
input_3
PythonBridge_1
DataViewer_5
input0
isStop

```
elif self.averageStop == 1 :
    print(f"============================STOP==================================")
    self.arretStop = 1
    self.distanceArret = (self.currentspeed * 0.1)**2 + 0.1 * (self.currentspeed * 1000/3600) * 100
    print(f"distanceArret :  {self.distanceArret}, distanceStop : {self.distanceStop}")
    if self.distanceStop <= self.distanceArret: #frein absolu
        print("frein 100% URGENCE")
        self.brake = 2.
    elif self.distanceStop > self.distanceArret and self.currentspeed > 7: #Si j'ai de la marge pour freiner
        print(f"FREIN SELON COURBE")
        # self.brake = (lambda x, a, b, c: a*x**2 + b*x + c)((self.distanceStop-self.distanceArret), *self.params2)
        self.exceptspeed = 5.
        if self.currentspeed <=10:
            self.brake = 0.6
        elif self.currentspeed <= 12:
            self.brake = 0.85
        elif self.currentspeed <= 15:
            self.brake = 0.95
        elif self.currentspeed <= 20:
            self.brake= 1.10
        elif self.currentspeed <=24:
            self.brake = 1.25
        elif self.currentspeed <=30:
            self.brake = 1.55
        else :
            self.brake = 1.8
        self.arret = 1
        print(f"frein : {self.brake}")
    else : #si j'ai bien freiné
        self.brake = 0.0
        # self.averageStop = self.inputs["averageStop"].ioelt.data
        print(f"AVANCE JUSQUA STOP")
        self.brake = 0.0
        self.exceptspeed = 5.
        # self.averageStop = self.inputs["averageStop"].ioelt.data
        self.arret = 1
        # if ((self.distanceStop-self.distanceArret) >=800):
        #     print("frein 0%")
        #     self.brake=0.
        # else :
        #     self.brake = exponential_function((self.distanceStop-self.distanceArret), *self.params)
        #     print(f"frein : {self.brake}")
    #Arret
    # self.brake = max(self.brake, self.brakeObs)
    self.outputs["brake"].write(self.brake*1.0)
    self.outputs["speed"].write(self.exceptspeed)
```

Dans ce code si un stop est détecté, nous calculons notre distance d'arrêt (distance parcourue pendant le temps de réaction + distance de freinage).

Si la distance à laquelle se trouve le panneau stop est inférieur à la distance d'arrêt, nous appliquons 100% du frein.

Sinon si la distance est supérieure et que nous sommes à plus de 7km/h, nous appliquons du frein dépendant de notre vitesse.

Sinon si la vitesse est inférieure à 7km/h et que la distance de freinage est suffisamment grande, nous avançon jusqu'au stop à 5km/h.

De plus nous passons une variable à 1 indiquant que nous devons nous arrêtez.

```
if self.averageStop == 0 and self.arret == 1:
    self.outputs["brake"].write(1.0)
    self.outputs["speed"].write(0.0)
    print(f"============================ARRET==================================")
    time.sleep(3)
    self.outputs["brake"].write(0.0)
    # self.outputs["speed"].write(self.max_speed)
    self.arret = 0
    self.exceptspeed = self.max_speed
    self.speed =0.
```

Finalement, si nous ne voyons plus de stop (signifie que l'on est à moins de 2 mètres car hors champ), et que la variable d'arrêt est à 1, nous procédons à l'arrêt pendant 3 secondes.