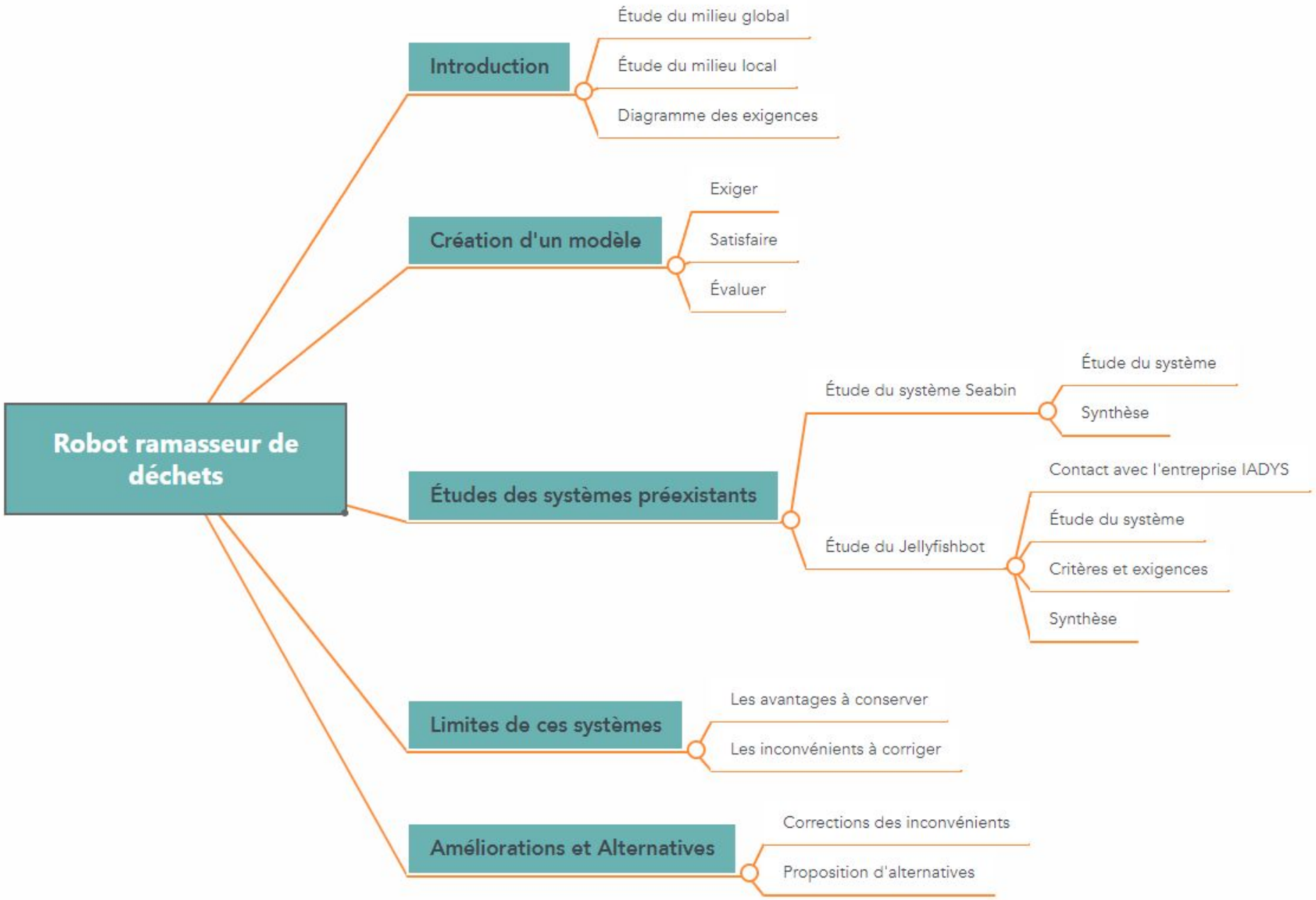
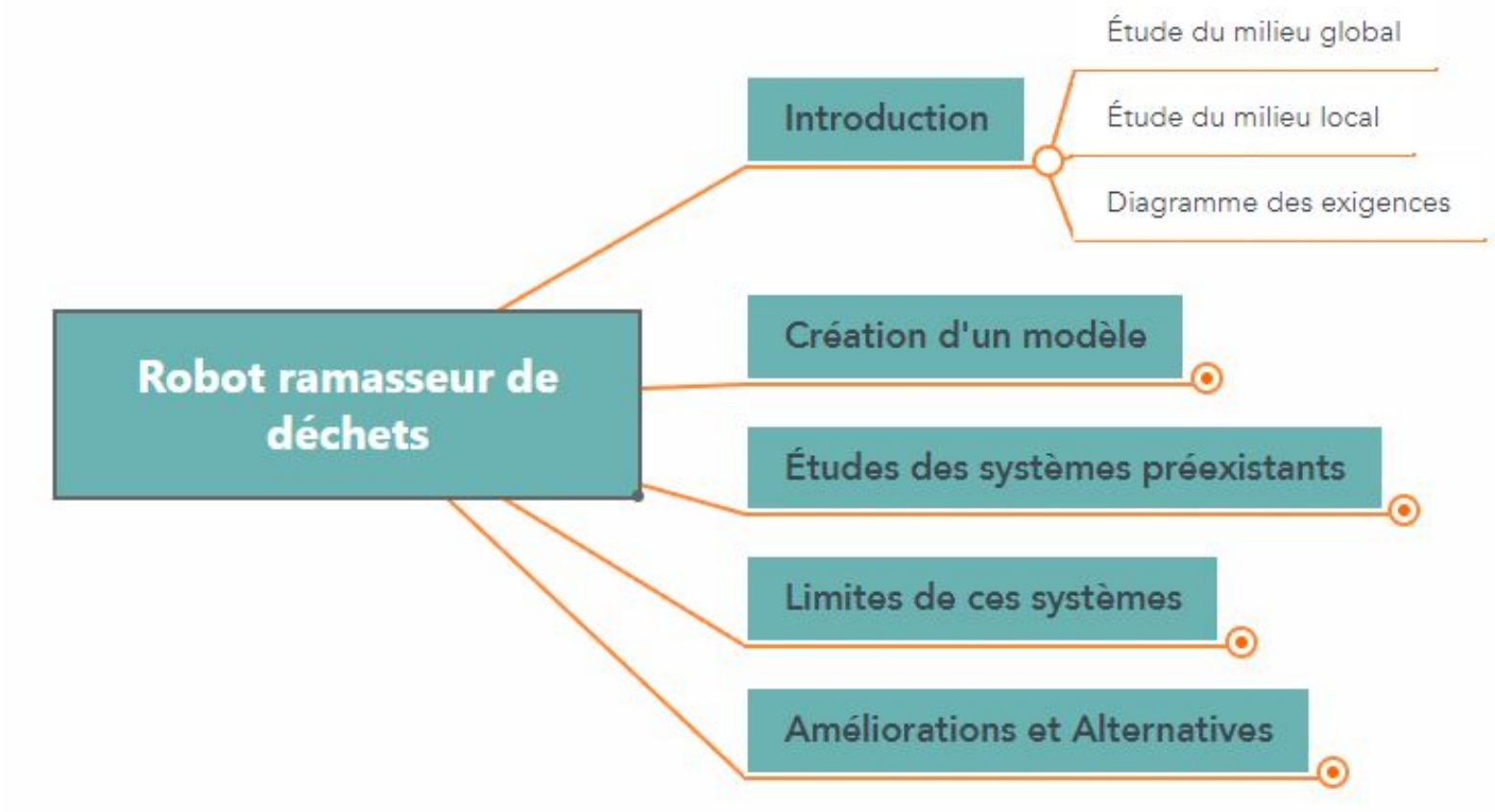
Robot ramasseur de débris plastiques flottants

Étude, modèle et améliorations

Professeurs encadrants : Olivier GRAS, Laurent BETSCHART


Robot ramasseur de déchets
Introduction
Étude du milieu global
Étude du milieu local
Diagramme des exigences
Création d'un modèle
Exiger
Satisfaire
Évaluer
Études des systèmes préexistants
Étude du système Seabin
Étude du système
Synthèse
Étude du Jellyfishbot
Contact avec l'entreprise IADYS
Étude du système
Critères et exigences
Synthèse
Limites de ces systèmes
Les avantages à conserver
Les inconvénients à corriger
Améliorations et Alternatives
Corrections des inconvénients
Proposition d'alternatives


Robot ramasseur de déchets
Introduction
Étude du milieu global
Étude du milieu local
Diagramme des exigences
Création d'un modèle
Études des systèmes préexistants
Limites de ces systèmes
Améliorations et Alternatives

Étude du milieu

Mise en place du cadre de l' étude

Division de l' étude en deux parties :

- milieu global : étude des caractéristiques générales du milieu, indépendantes de la position du robot.
- milieu local : étude des caractéristiques de l' environnement proche pouvant varier selon la position et le temps.

L' étude du milieu est nécessaire puisqu'elle permet, au-delà de poser le cadre de l' étude, de dégager certaines exigences que le système doit satisfaire. Ces exigences serviront donc à valider ou non la pertinence du modèle retenu et des systèmes préexistants étudiés.

Milieu global

Milieux considérés : Milieu aquatique

Caractéristiques	Description
Température	Température supposée constante du milieu dans l' eau et en surface
Qualité de l' eau	Prise en compte de la teneur en hydrocarbure de l' eau
Bathymétrie	Topographie des profondeurs
Mouvements	Prise en compte des courants, de la houle ou de perturbations annexes
Météorologie	Étude de la météorologie et de ses conséquences sur le système, directs et indirects.

Milieu local

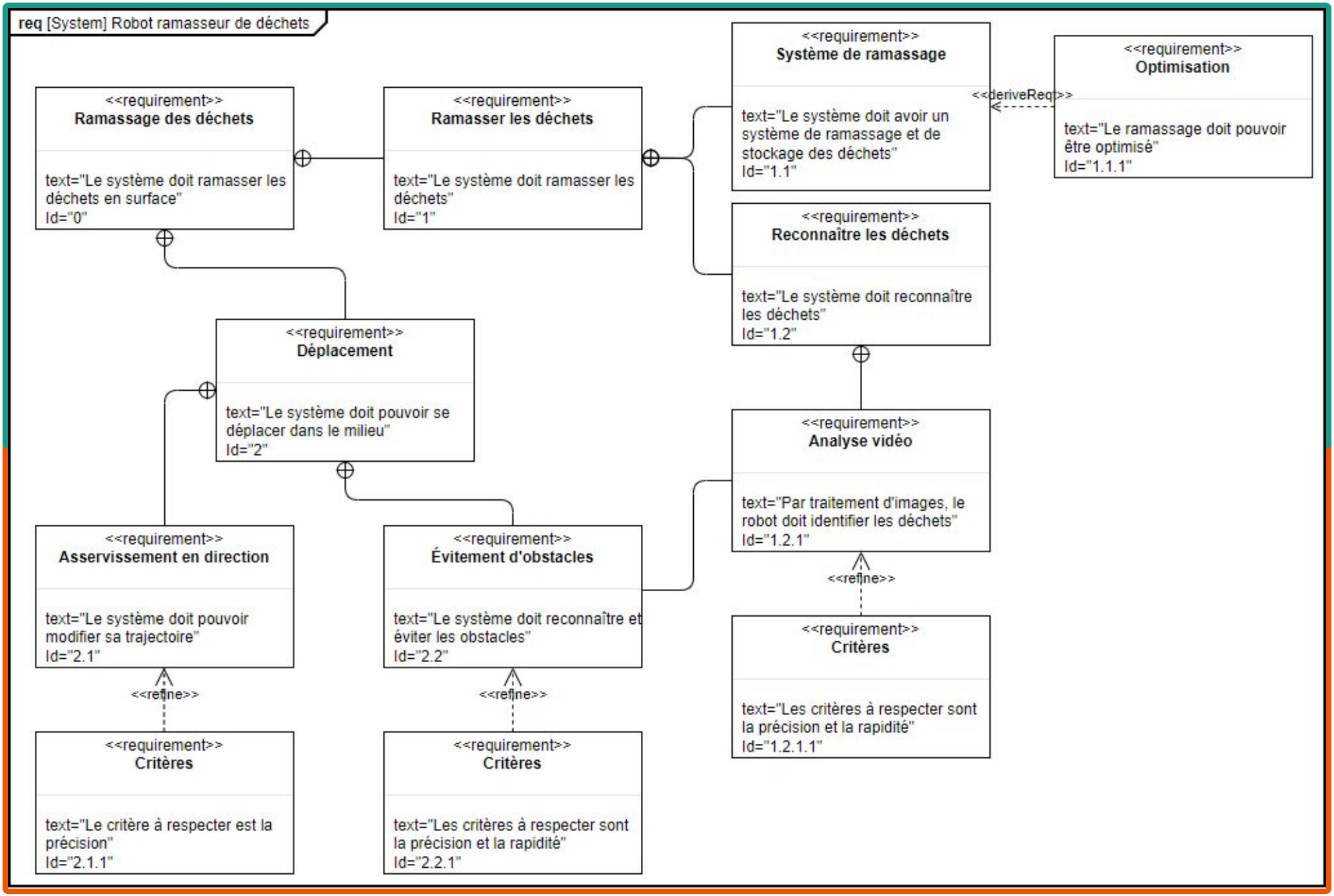
Étude de la proximité du système dans le milieu global défini précédemment

Caractéristiques	Description
Biodiversité	Prise en compte des animaux pouvant interagir avec le système
Déchets	Prise en compte des déchets pouvant interagir avec le système

Diagrammes des exigences

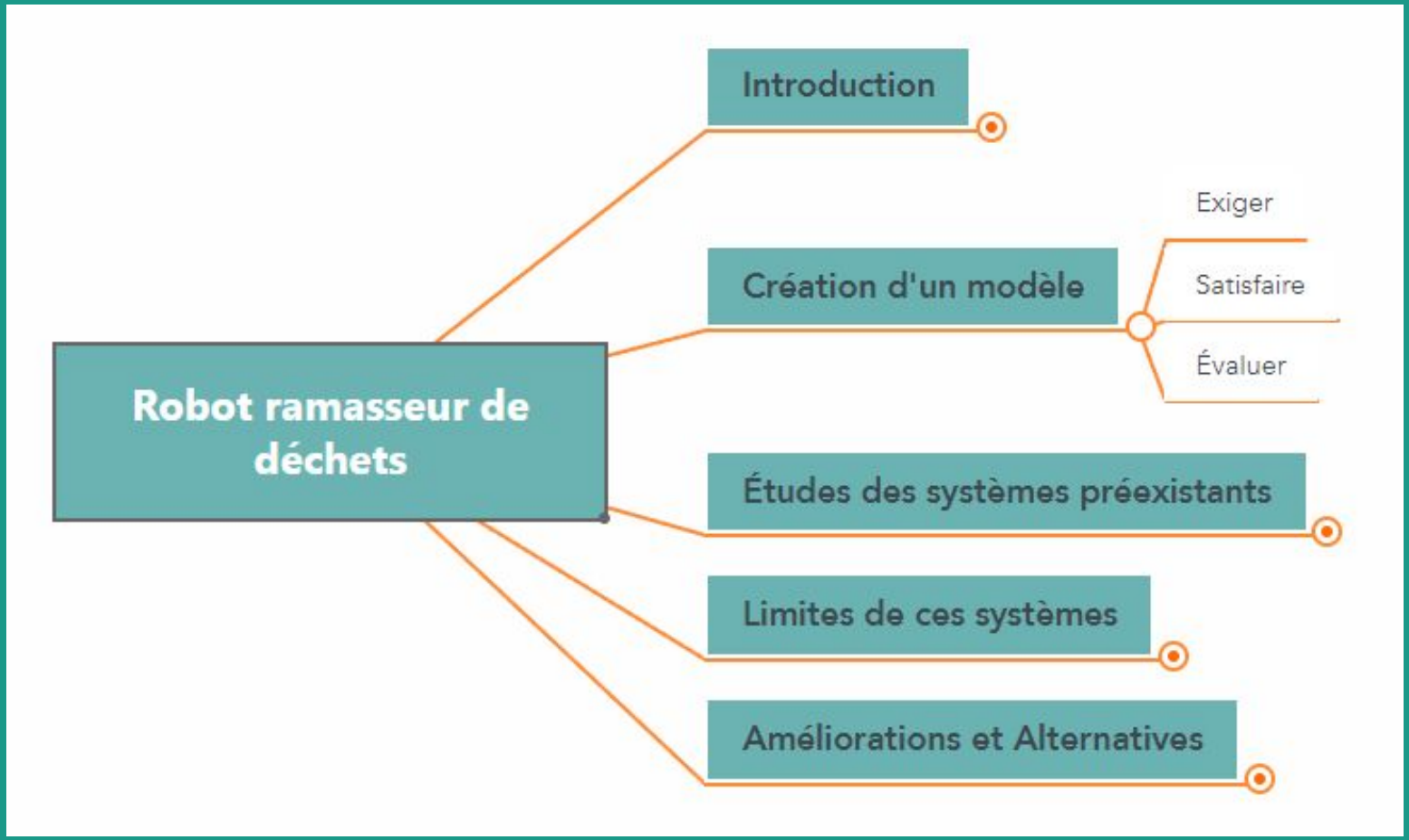
Mise en place des attentes du système

L' étude du milieu faite précédemment et en prenant en compte les besoins que le système doit satisfaire, nous pouvons synthétiser les exigences attendues en un diagramme SysML normalisé du type requirement diagram.



Source : réalisation personnelle

Création d'un modèle



Exiger

I. Les objectifs

Les objectifs du système sont :

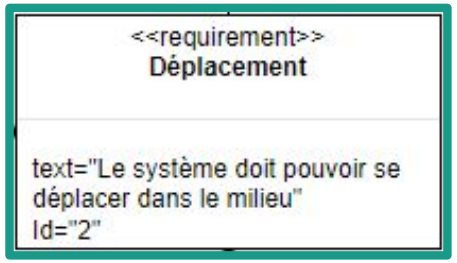
- **ramasser les déchets**
- **se déplacer dans le milieu**

Exiger

II. Les fonctions

Les exigences imposent la création de fonctions concernant:

- **le déplacement**



- **le ramassage**

Exiger

III. Les contraintes

Les contraintes imposées sont relatives au milieu et aux objectifs du système. Concernant le milieu, le robot devra éviter les différents obstacles qui risquent d'endommager le système. Concernant les objectifs, il faudra implémenter différentes solutions afin d'obtenir des résultats satisfaisants.

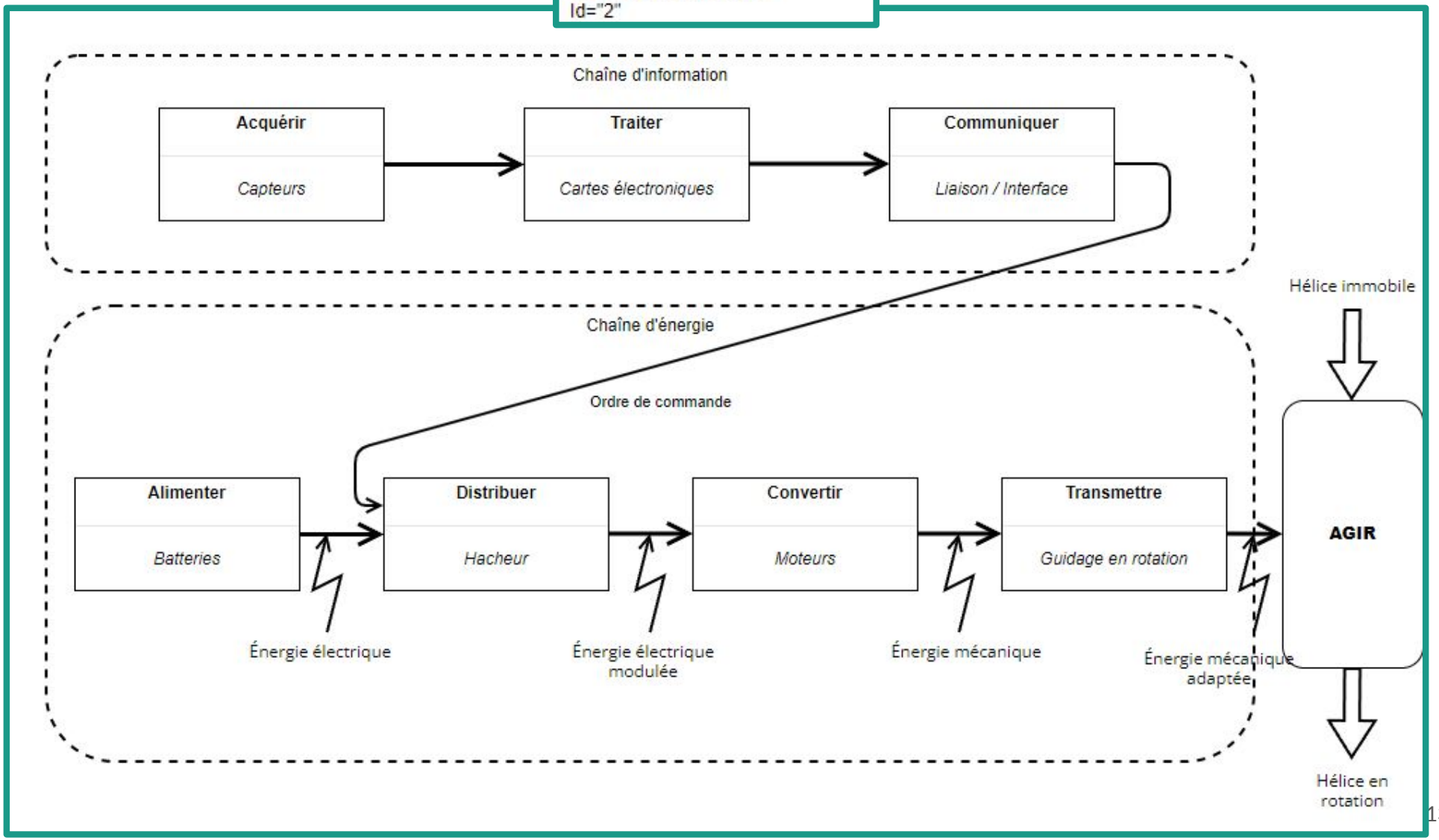
Satisfaire

I. Créer les fonctions

<<requirement>>
Déplacement

text="Le système doit pouvoir se déplacer dans le milieu"
Id="2"

Source : réalisation personnelle

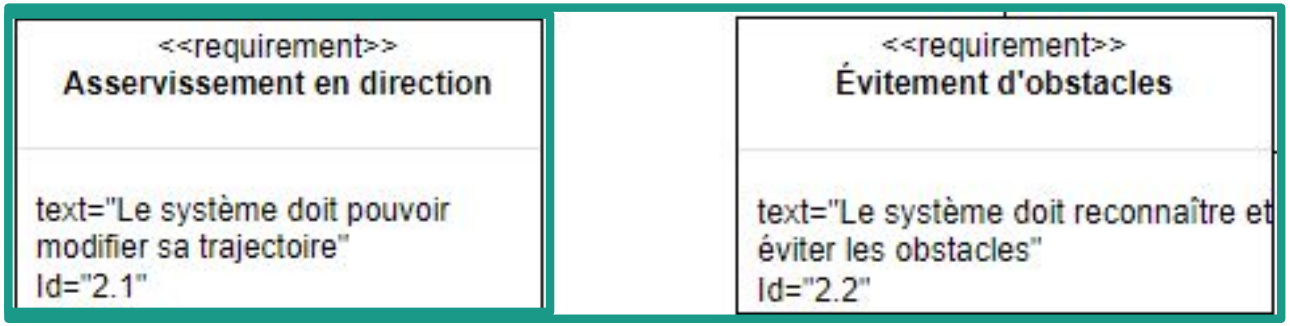


Satisfaire

II. Résoudre les contraintes

I. Éviter les obstacles

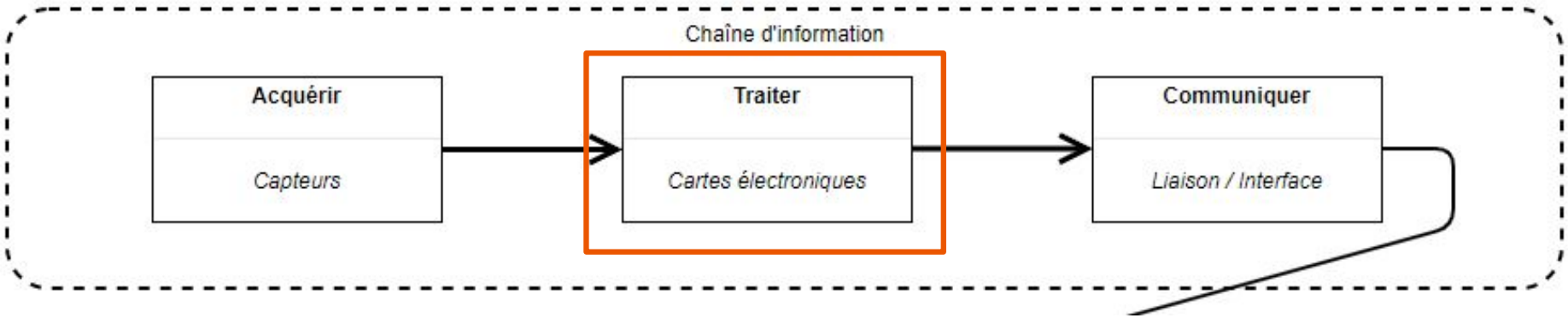
II. Remplir les conditions



<<requirement>>
Évitement d'obstacles

text="Le système doit reconnaître et éviter les obstacles"
Id="2.2"

Évitement d'obstacles : méthode informatique



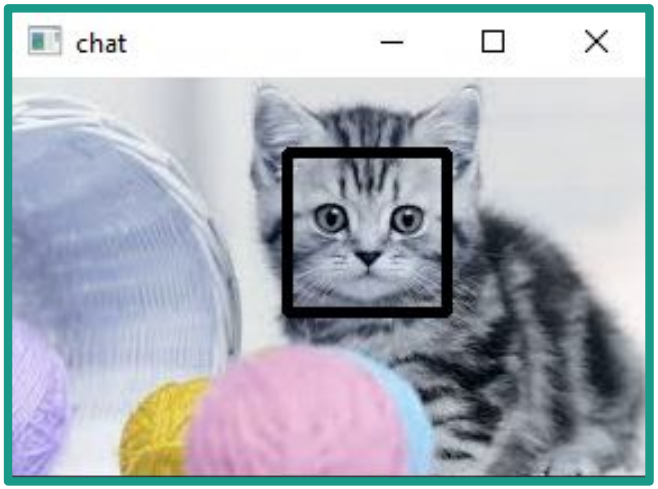
Pour mettre en place l' évitement d'obstacle, on agit directement sur la chaîne d'information. On recueille les informations à l' aide des capteurs et on met en place un algorithme de traitement d'images directement dans la carte électronique (ou dans le système responsable du traitement de l' information).

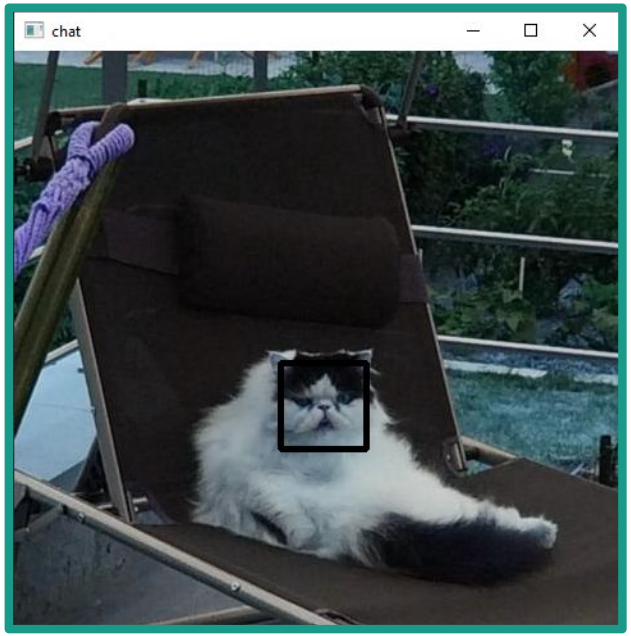
Algorithme de traitement d'images

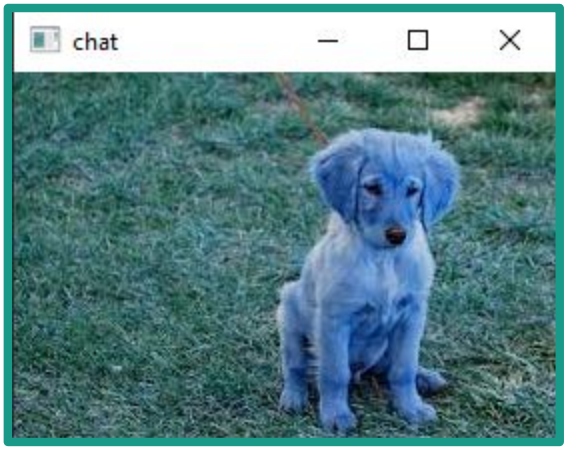
détection_tipe.py

```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

img_bgr = cv.imread("chat.jpg", 1)
img_rgb = cv.cvtColor(img_bgr, cv.COLOR_BGR2RGB)

faceCascade = cv.CascadeClassifier("haarcascade_frontalcatface.xml")
detectedFaces = faceCascade.detectMultiScale(img_rgb, 1.1, 4)
print(detectedFaces)
for (longueur, largeur, width, hauteur) in detectedFaces:
    cv.rectangle(img_rgb, (longueur, largeur), (longueur + width, largeur + hauteur), (0, 0, 0), 4)
cv.imshow('chat',img_rgb)
cv.waitKey(0)
```

Résultats de l' algorithme

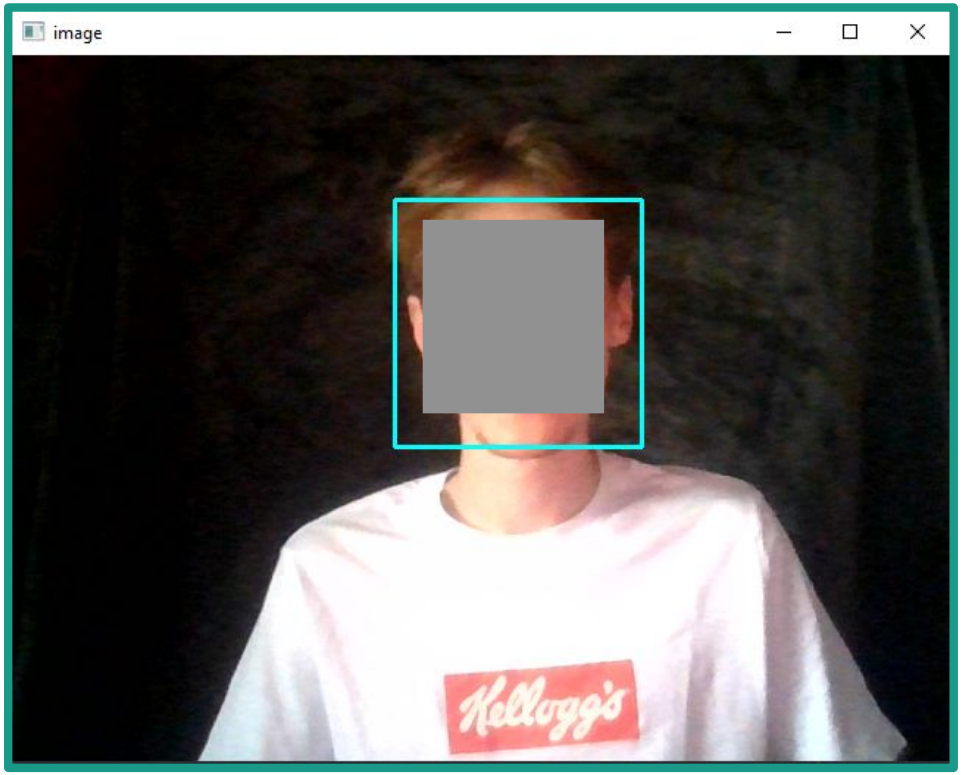






Application de l' algorithme au direct

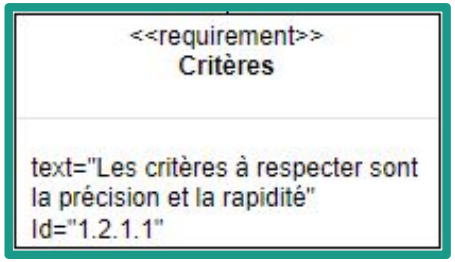
```python
while(True):
    t0 = time.time() #on récupère le temps avant la détection
    test, image = direct.read() #direct.read() est un tupple composé d'un test puis de l'image
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) #on transforme l'image en niveau de gris
    visages = visage_cascade.detectMultiScale(gray, 1.3, 5) #on effectue la détection de visage
    for (x, y, largeur, hauteur) in visages:
        cv2.rectangle(image, (x, y), (x+largeur, y+hauteur, (255, 255, 0), 2) # on trace un rectangle autour du visage détecté
    cv2.imshow('image', image) #on affiche le résultat de l'algorithme
    k = cv2.waitKey(1)
    liste_temps.append(time.time()-t0) #on stocke dans une liste le temps de la détection
    if k == ord('q') or k == 27:
        break
```




```
moyenne du temps de détection = 0.043987743942825888
```

Détection réalisée à l' aide d'OpenCV sur Python

Changement pour un algorithme de reconnaissance de visage, plus simple à mettre en place en pratique pour un flux vidéo en direct



Critiques et observations

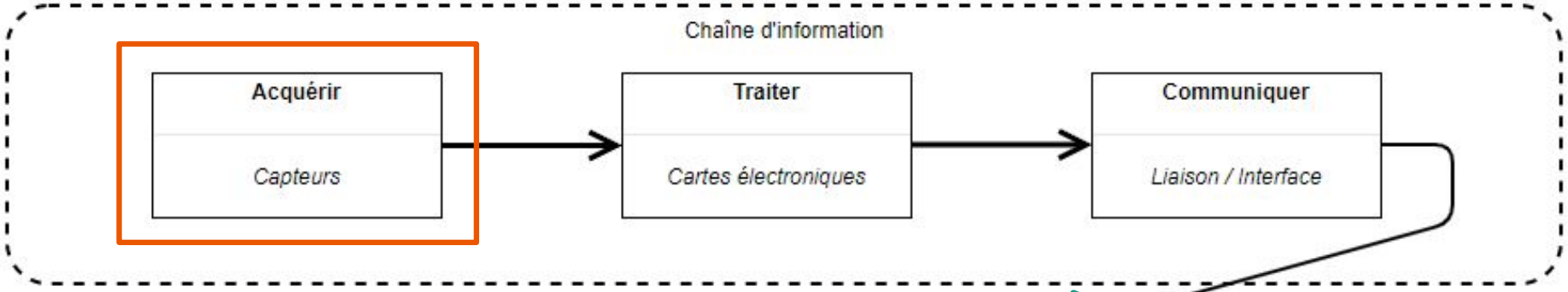
Avantages	Critiques
Rapide	Dépend énormément de conditions telles que la lumière, l' angle de vue ou la qualité de la caméra
Assez précis	Risque d’erreurs et incertitudes assez grandes

Cependant, cette méthode fait aussi émerger de nombreux problèmes

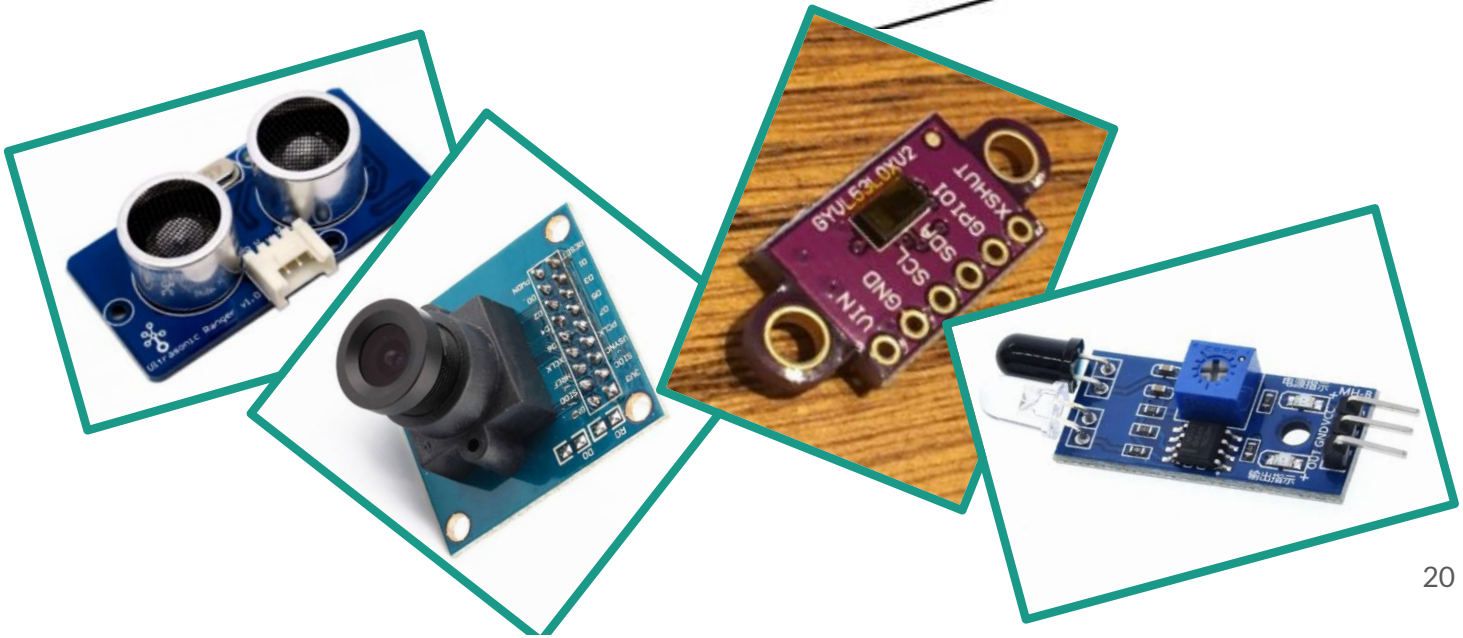
<<requirement>>
Évitement d'obstacles

text="Le système doit reconnaître et éviter les obstacles"
Id="2.2"

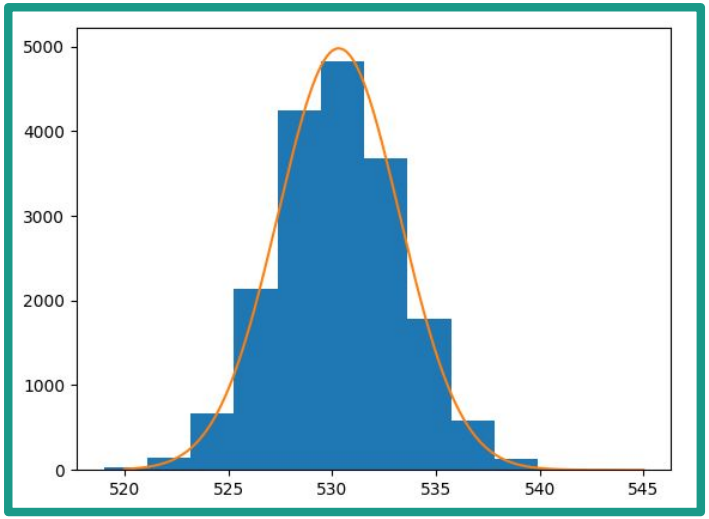
Évitement d'obstacles : utilisation de capteurs



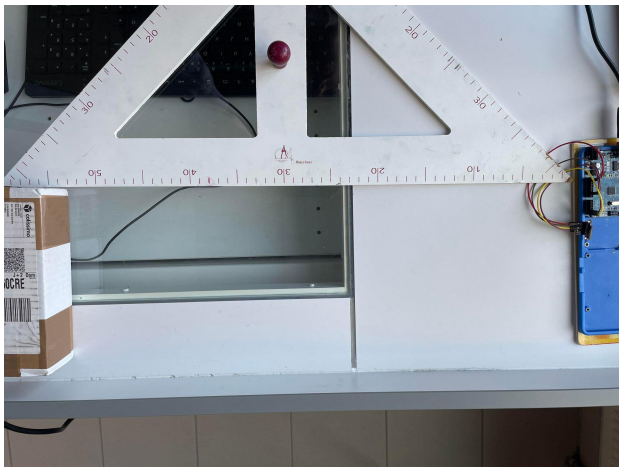
- LIDAR
- Ultrason
- Caméra
- Infrarouge

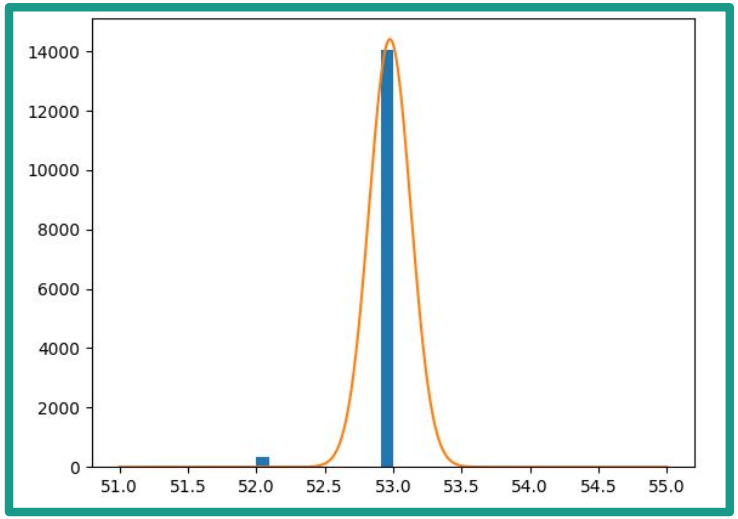


Exemples de capteur : LIDAR/Ultrasonore

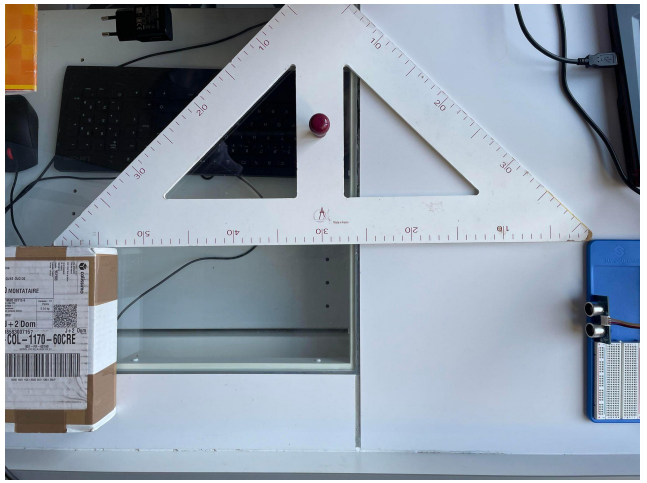


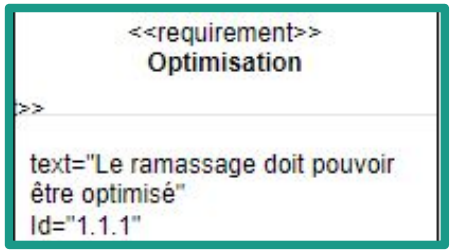
18 251 mesures, résolution de 1 mm





14 397 mesures, résolution de 1 cm





Remplir les conditions

Pour remplir l' objectif, améliorer les résultats et optimiser le ramassage, plusieurs techniques sont possibles.

<<requirement>>
Asservissement en direction

text="Le système doit pouvoir modifier sa trajectoire"
Id="2.1"

Fonctions informatiques de direction

```python
import cv2
from pyfirmata import Arduino, SERVO
from time import sleep

carte = Arduino("COM6")
pin = 8
carte.digital[8].mode = SERVO
carte.digital[8].write(180)

L=[]
i = 0
def rotationservo(pin, angle):
    carte.digital[pin].write(angle)
    sleep(0.015)


def rotation(pin):
    for i in range(0,180):
        rotationservo(pin, i)
    for j in range(180,1,-1):
        rotationservo(pin,j)
```

Rendement et intelligence artificielle

Stratégies :

- Suivre le déchet le plus proche
- Identifier tous les déchets présents dans l' environnement et appliquer l' algorithme du plus court chemin

<<requirement>>
Asservissement en direction

text="Le système doit pouvoir modifier sa trajectoire"
Id="2.1"

Suivre le déchet le plus proche

```python
import cv2
import numpy as np
import time
from pyfirmata import Arduino, SERVO

carte = Arduino("COM5")
pin = 7
carte.digital[7].mode = SERVO
carte.digital[7].write(0)
L=[]

liste_temps = []

direct = cv2.VideoCapture(0)
direct.set(3, 640)  # largeur
direct.set(4, 480)  # hauteur

visage_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_alt.xml")
```

Suivre le déchet le plus proche

```
while(True):
    t0 = time.time() #on récupère le temps avant la détection
    test, image = direct.read() #direct.read() est un tupple composé d'un test puis de l'image
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) #on transforme l'image en niveau de gris
    visages = visage_cascade.detectMultiScale(gray, 1.3, 5) #on effectue la détection de visage
    for (x, y, largeur, hauteur) in visages:
        cv2.rectangle(image, (x, y), (x+largeur, y+hauteur), (255, 255, 0), 2) #on trace un rectangle autour du visage détecté
    cv2.imshow('image', image) #on affiche le résultat de l'algorithme
    (dim_x,dim_y,c)=image.shape # on récupère la dimension de l'image
    center_x=dim_x//2 # on travaille sur le centre de l'axe horizontal
    for i in range(len(visages)):
        L.append(abs((visages[i][0]+largeur)/2-center_x))
    if len(L)!= 0: #on sélectionne l'objet le plus proche du centre
        m = min(L)
        j = L.index(m) #on stocke son indice et on travaille dessus
        if (visages[j][0]+largeur)/2-center_x>10: #si l'objet se trouve à droite du centre
            print('right',(visages[j][0]+largeur)/2-center_x)
            carte.digital[pin].write(135) #on modifie la direction
        elif (visages[j][0]+largeur)/2-center_x<-10: #si l'objet se trouve à gauche du centre
            print('left', (visages[j][0]+largeur)/2-center_x)
            carte.digital[pin].write(45)  #on modifie la direction
        else: #si l'objet n'est pas éloigné du centre
            print('forward')
    del L[::] #on vide la liste des objets détectés pour éviter leur accumulation
    k = cv2.waitKey(1)
    liste_temps.append(time.time()-t0) #on stocke dans une liste le temps de la détection
    if k == ord('q') or k == 27:
        break
```

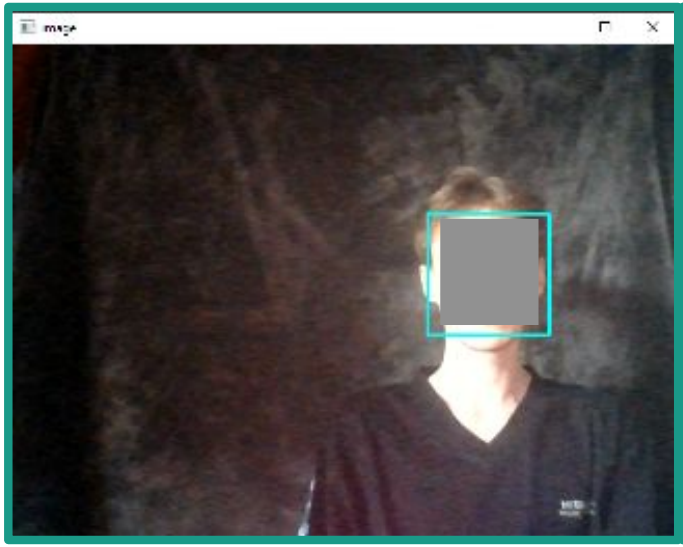
détection

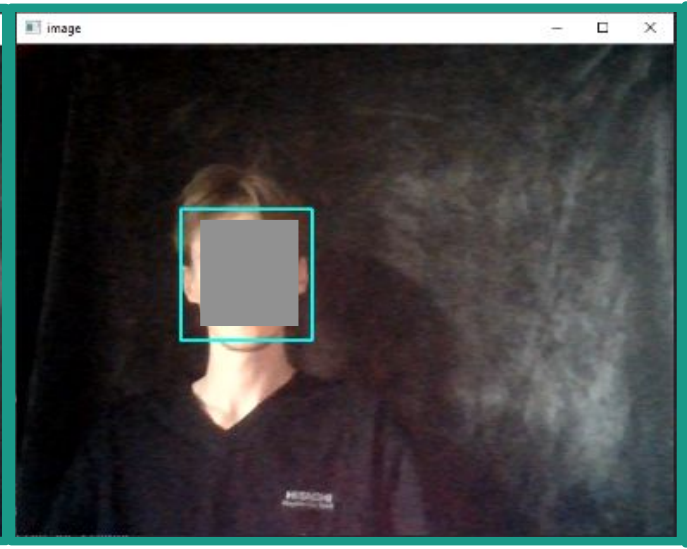
étude de la position relative par rapport au centre de l' image

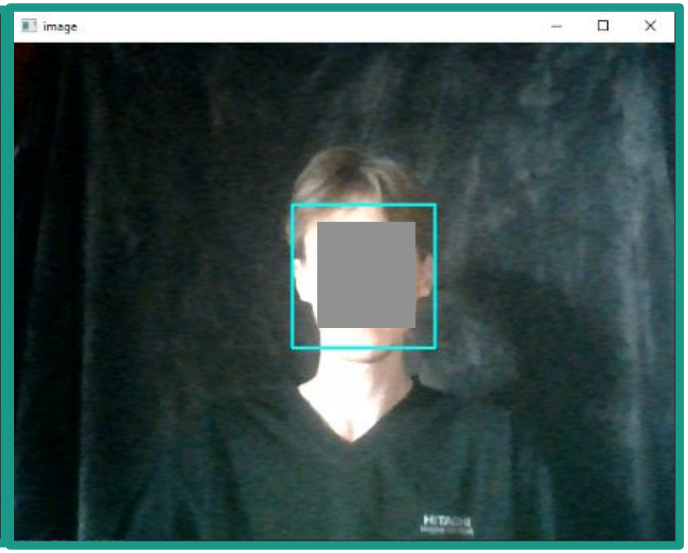
Résultats expérimentaux

0°

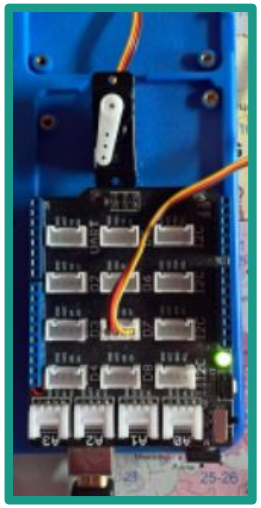
180°



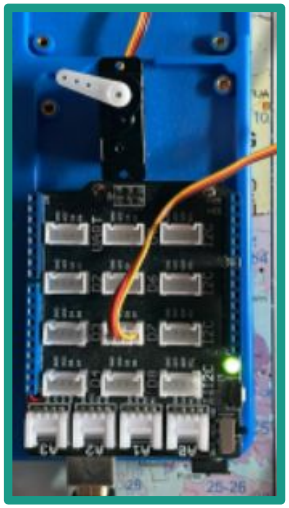




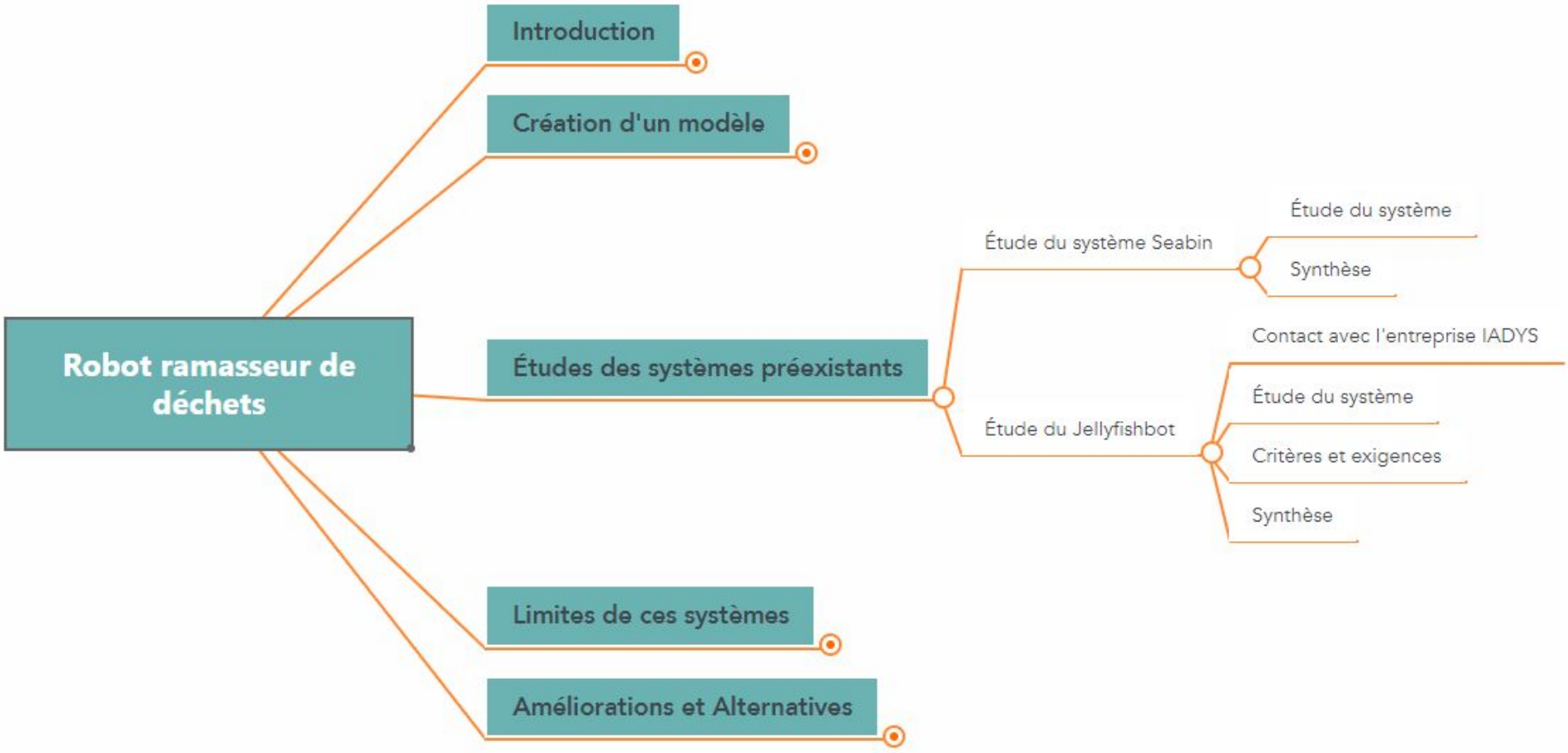

```
right 20.5
right 20.5
right 20.0
right 20.0
right 21.5
right 21.5
right 21.0
right 21.0
```



```
left -97.0
left -97.0
left -95.0
left -95.0
left -95.0
left -95.0
left -94.5
left -94.5
```



```
forward
forward
forward
forward
forward
forward
forward
forward
```


Robot ramasseur de déchets
Introduction
Création d'un modèle
Études des systèmes préexistants
Étude du système Seabin
Étude du système
Synthèse
Étude du Jellyfishbot
Contact avec l'entreprise IADYS
Étude du système
Critères et exigences
Synthèse
Limites de ces systèmes
Améliorations et Alternatives

Étude des solutions préexistantes

Après avoir posé le cadre de l' étude et définit le modèle, on étudie les solutions préexistantes

1. Seabin
    1.1. Étude du système
    1.2. Synthèse

2. Jellyfishbot
    2.1. Contact avec l' entreprise
    2.2. Étude du système
    2.3. Critères et exigences
    2.4. Synthèse

I. Seabin

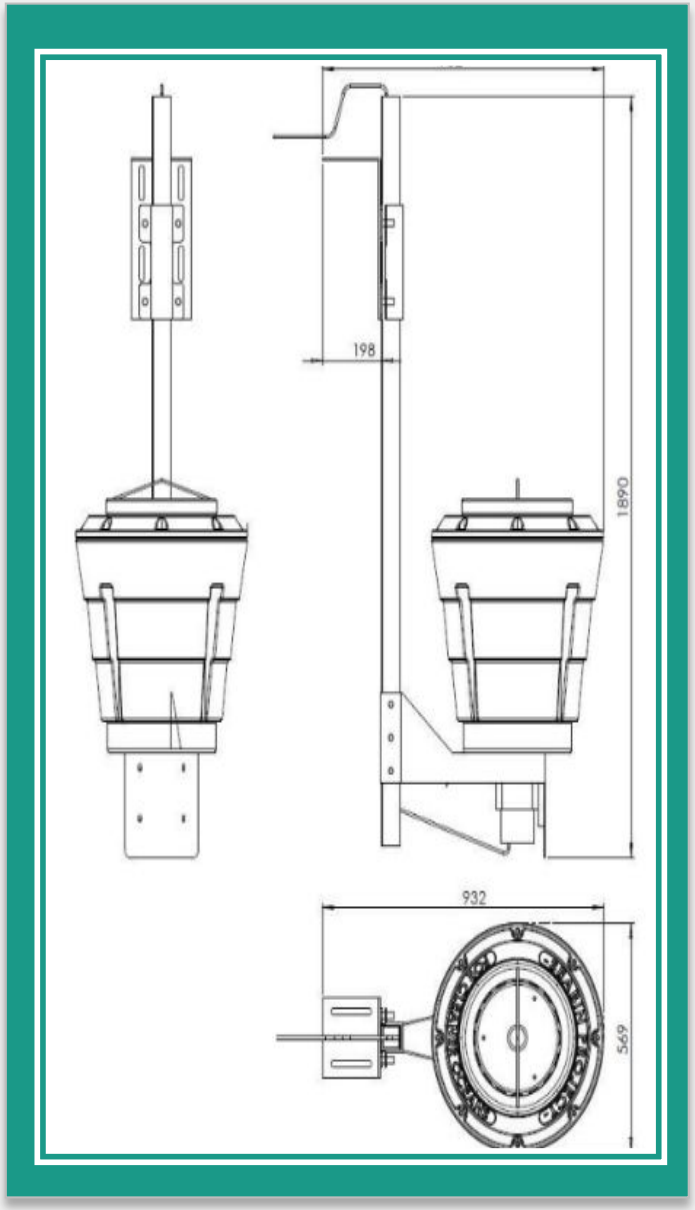
Étude du système

Dimensions : 50 x 50 x 180 cm

Masse (avec support et sans déchets) : 55 kg

Caractéristiques du ramassage :

- Contenance du sac : 20 kg
- Peut ramasser des déchets d'un diamètre supérieur à 2mm



Source :
https://seabinproject.com/the-seabin-v5/technical-specs/

Étude du système

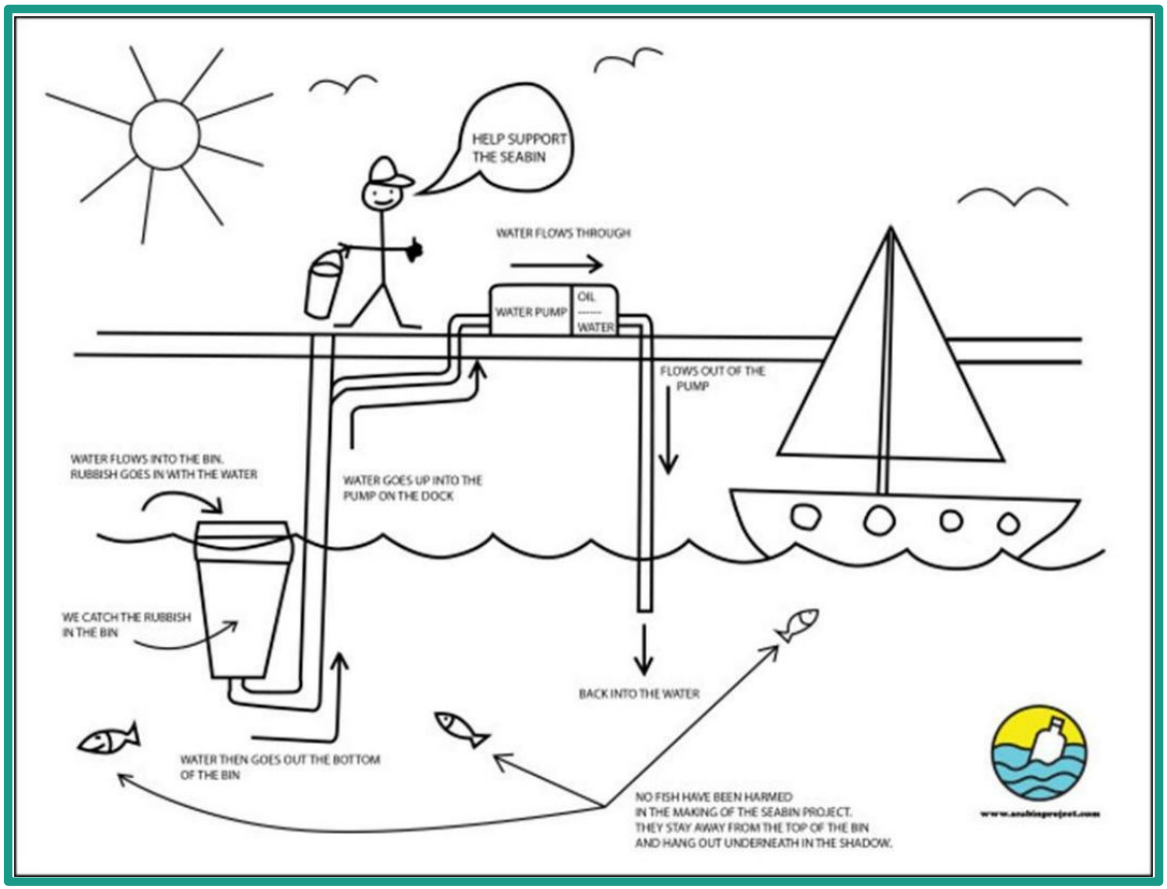
Système passif

Doit être entretenue quotidiennement

Fonctionnement à l' aide du pompe

Caractéristiques de la pompe :

- Power: 110V / 220V Consumption 2.5 amps 500 watts
- Pump: 25,000 LPH

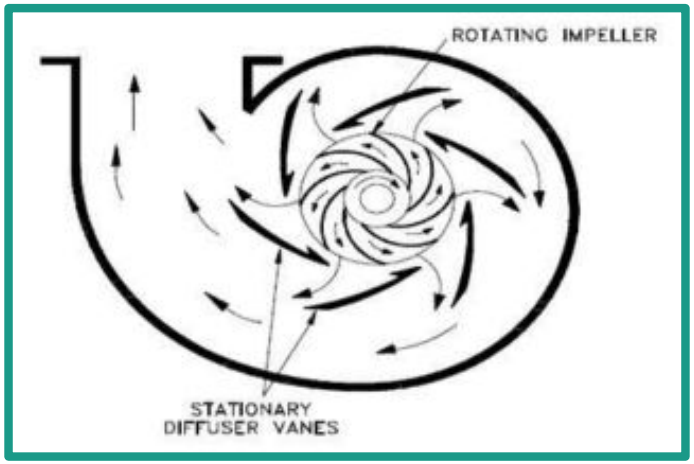


Étude de la pompe

Par manque d'informations techniques sur le système technique, on étudie une pompe dont le fonctionnement et les caractéristiques sont identiques.

Référence : SP032022A03-01

Type de fonctionnement : Pompe centrifuge



Synthèse

Résultats attendus du système Seabin :

- Estimation : 1,5 tonnes de déchets récupérés par an

- Expérimentalement : 1,319 tonnes de déchets récupérés par an

Sources : https://seabinproject.com/the-seabin-v5/technical-specs/

II. Jellyfishbot

Contact avec l' entreprise IADYS

Pauline Thévenot <pauline.thevenot@iadys.com>
À Olivier, Amandine, moi

ven. 27 mai 09:40 (il y a 10 jours)

Bonjour Monsieur Gras,

Je vous remercie pour votre message et votre démarche.

Veuillez trouver ci-joint la data sheet de notre Jellyfishbot et de ses accessoires. Je pense que l'ensemble de vos questions sont traitées dedans, mais n'hésitez pas à revenir vers moi s'il vous fallait des informations complémentaires.

Bien à vous,

Pauline

iADYS
Interactive Autonomous DYnamic Systems

Pauline Thévenot
RESPONSABLE COMMUNICATION & MARKETING
COMMUNICATION & MARKETING MANAGER

+33 (0)7 63 15 11 25
Immeuble LIBER 1 - N°Z09
ZA Le clos du rocher,
13830 Roquefort-la Bédoule
www.iadys.com – www.jellyfishbot.io

Confidentiality Notice

This e-mail, including any attachments and response string, may contain proprietary information which is confidential and may be legally privileged. It is for the intended recipient only. If you are not the intended recipient or transmission error has misdirected this e-mail, please notify the author by return e-mail and delete this message and any attachment immediately. If you are not the intended recipient you must not use, disclose, distribute, forward, copy, print or rely on this e-mail in any way except as permitted by the author.

Étude structurelle

Dimensions : Le Jellyfishbot mesure 70 x 70 x 50 cm

Masse : Le robot pèse 20 kg (hors déchets ramassés)

Caractéristiques du ramassage : Le Jellyfish bot possède différents filets d'une contenance totale de 80 litres, permettant de collecter des déchets de 150 µm à 30cm. Les filets sont aussi des absorbants pour les hydrocarbures : jusqu'à 30 litres de contenance.

Source : https://www.jellyfishbot.io

Caractéristiques du ramassage

Filets

	Filet macro-déchets jetable	Filet upcyclé macro-déchets réutilisable	Filet milli-déchets réutilisable	Filet micro-déchets réutilisable
Taille du maillage	5 à 6 mm	10 à 15 mm	1 mm	150, 180, 250, 300 µm
Capacité	80 L	80 L	70 L	70 L
Unité de vente	100	1	1	1
Usages	Macro-déchets, hydrocarbures	Macro-déchets	Billes de polystyrène, lentilles d'eau, etc.	Microplastiques, poussières de peinture, etc.
				

Matériaux absorbants d'hydrocarbures

	Spaghetti	Feuilles	Barrages à jupe	
Capacité de collecte	> 30 L par filet	> 1 L par feuille	30 à 80 L pour 3 m	60 à 165 L pour 6 m
Taille	0,5 cm x 8 cm	40 cm x 50 cm	⌀ 20 cm x 3 m	⌀ 20 cm x 6 m
Unité de vente	5 kg	200 pcs	1 pc	1 pc
				

Source : contact avec l' entreprise

iADYS

Étude mécanique du robot

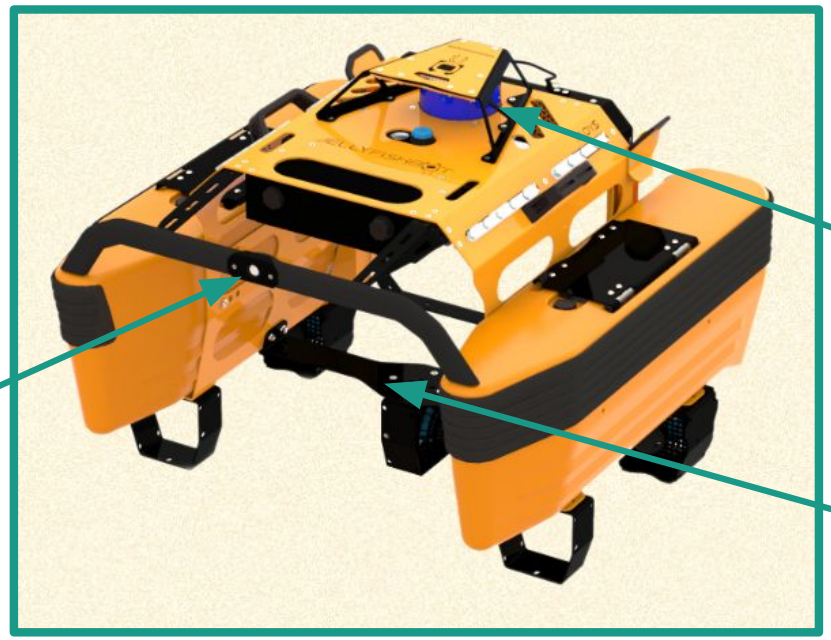
Mobilités : Le Jellyfishbot possède trois propulseurs lui permettant d'avancer, de reculer, de tourner sur place et de se déplacer latéralement

Vitesse : Le robot possède une vitesse moyenne de 1 nœud mais sa vitesse maximum est de 2 nœuds.

Étude électronique

Capteurs : le Jellyfishbot possède plusieurs capteurs :

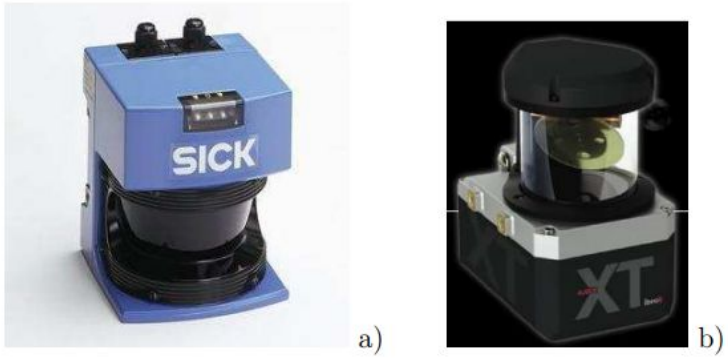
- LIDAR : utilisé dans la détection d'obstacle du robot autonome
- Sondes : permet de mesurer la qualité de l' eau ou d'effectuer des relevés bathymétriques
- Caméra embarquée : permet de piloter le robot à distance



Étude du capteur LIDAR

Avantages :

- Précis sur une longue portée (25 mètres de portée pour le Jellyfishbot et 270° d'angle de mesure)
- Peu sensible aux variations météorologiques
- Détecte n'importe quel obstacle tant que celui-ci renvoie une partie de la lumière incidente

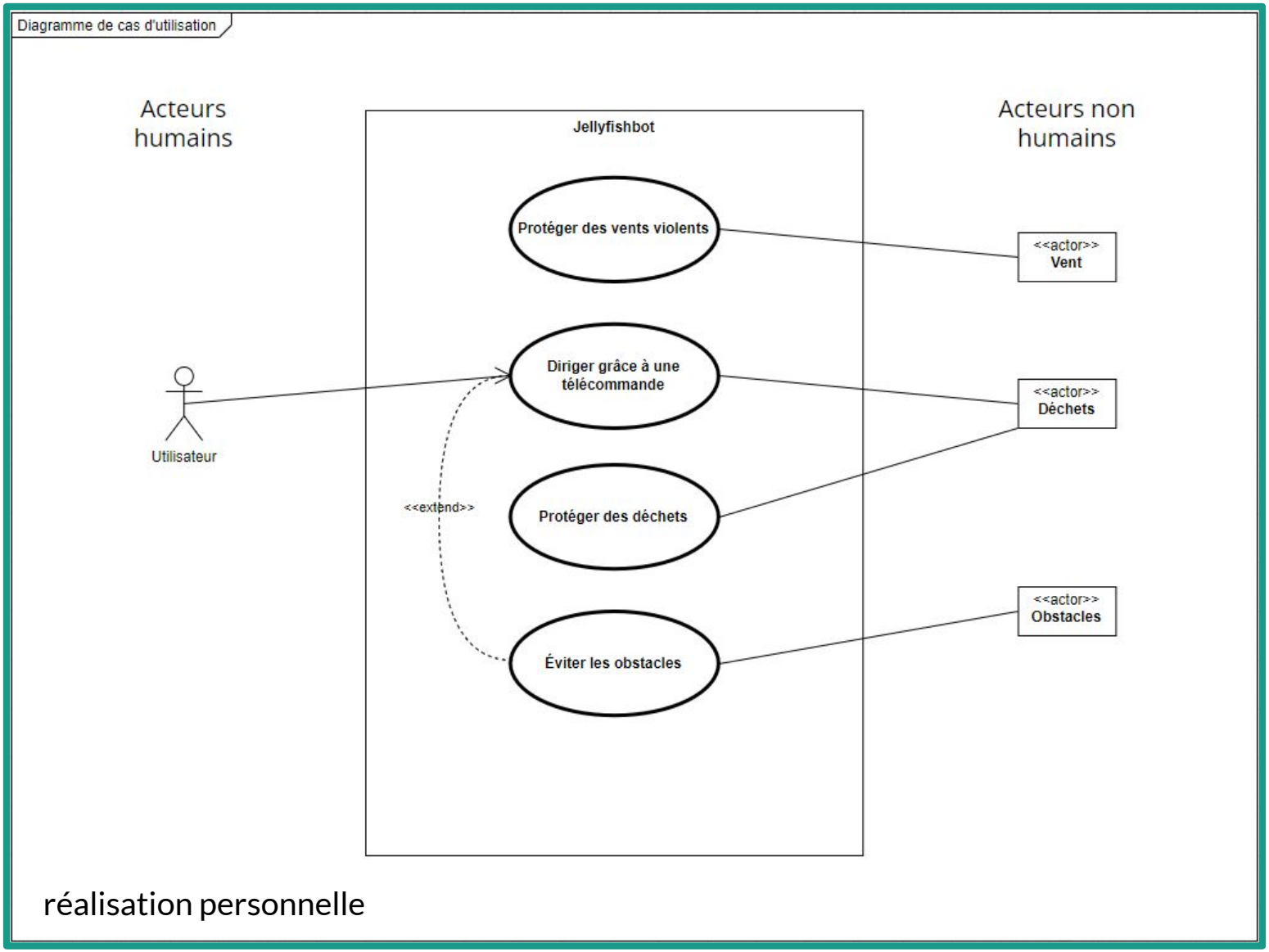


a) b)

FIG. 1.3 – *Deux lidars utilisés pour la détection d'obstacle. a) Un télémètre à balayage 2D générant un plan de coupe. b) Un télémètre à balayage 2D générant 4 plans.*

Étude du fonctionnement (non autonome)

Autonomie de 6 à 8h à l' aide de batteries rechargeables en 2h

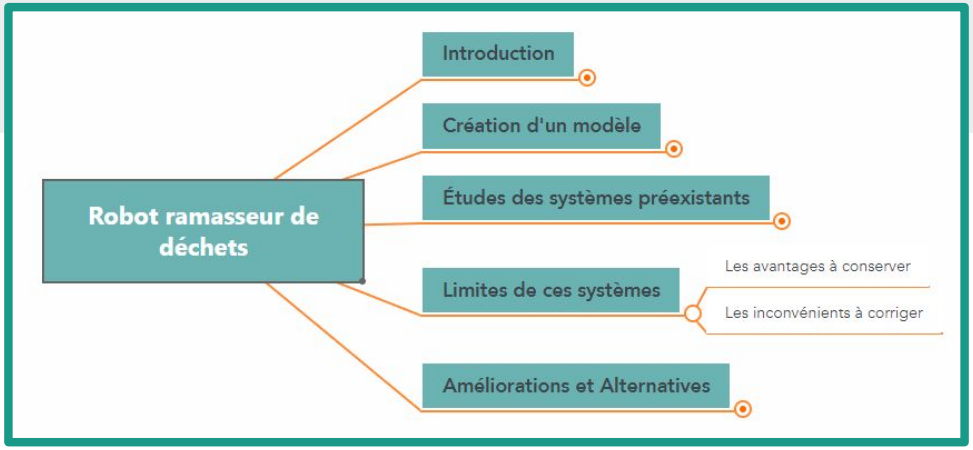


réalisation personnelle

Synthèse

Résultats attendus du système JellyfishBot :

- Résultats variables selon la concentration de déchets, le niveau de maîtrise de l' utilisateur…
- On peut cependant faire une estimation : pour un filet d'une contenance de 80 litres, on peut approximer que le montant amassé pour une moyenne d'une utilisation complète du Jellyfishbot par jour s'élève à environ 29 tonnes par an.



Limites de ces systèmes

Seabin :

Avantages :

- Nécessite peu d'entretien
- N'est pas affecté par les obstacles

Inconvénients :

- Compliqué à améliorer et à optimiser
- Doit être relié à la surface

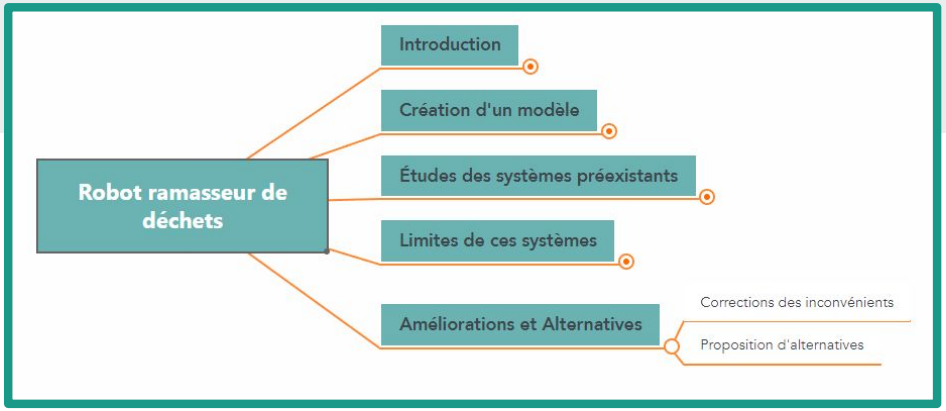
Jellyfishbot :

Avantages :

- Peut se déplacer même loin des terres
- Plus simple à améliorer et à optimiser

Inconvénients :

- Affecté par les obstacles et le milieu



Améliorations et alternatives

Améliorations :

En s'inspirant de ce qui a été étudié, on peut retenir différentes pistes d'améliorations :

- l' optimisation du ramassage à l' aide de différentes méthodes : électroniques, algorithmiques…
- Des filets pouvant absorber les hydrocarbures en surface.

Alternatives :

L' étude s'est porté sur des systèmes de petites et moyennes tailles, il est possible d'envisager une solution de ce type sur des bateaux ou des drones de tailles plus conséquentes.

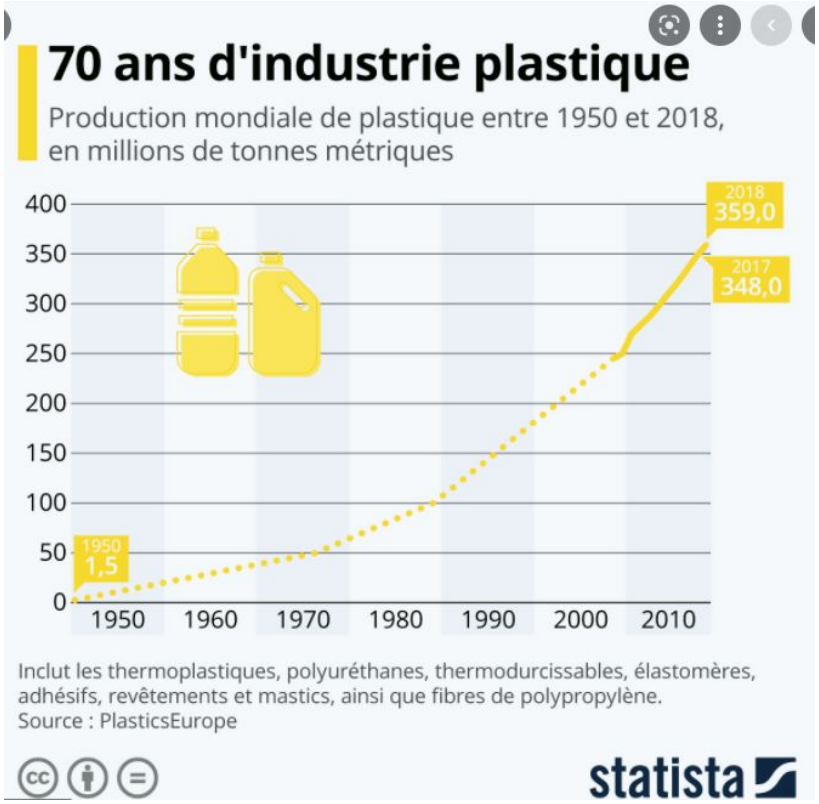
Annexe 1 : Matériaux possibles

Acier revêtu		Tuyautages, installations portuaires, carènes, boîtes à eau...
Alliage léger AG4MC	Mg 4 - Mn 0,4 - Cr 0,2	Carènes de petits navires.
Cupro-aluminium 1	Cu 87 - Al 9 - Ni 2 - Fe 2 - Mn 1,5	Pompes, hélices, plaques à tubes, tuyautage, vannes, boulonnerie...
Cupro-aluminium 2	Cu 81,5 - Al 9 - Ni 5 - Fe 3 - Mn 1,5	*idem* (meilleure résistance à la cavitation et à la corrosion sous tension).

Liste d'alliages utilisés dans l' eau de mer
Source : LA HOUILLE BLANCHE / N° 2/3 - 1973, J. Legrand -Études de matériaux susceptibles de résister à la corrosion par l' eau de mer

Sinon, le plastique reste le matériau résistant dans l' eau le moins coûteux, le plus léger et le plus malléable ce qui le rend pertinent dans notre cas.

Annexe 2 : Enjeux du TIPE



270 000 tonnes de plastique flottent à la surface des océans [1]

Les déchets plastiques occasionnent de nombreux dommages :

- accumulation dans les rivières et le long des infrastructures [2]
- Ces débris sont potentiels vecteurs pour des espèces animales et végétales invasives ce qui peut menacer nos ecosystèmes côtiers [1]

Liste des sources :

[1] https://hal.archives-ouvertes.fr/hal-03067254/document

[2] https://www.shf-lhb.org/articles/lhb/abs/2007/03/lhb2007041/lhb2007041.html

Annexe 3 : Proposition de valeurs numériques concernant le diagramme des exigences

Concernant le diagramme des exigences proposé, il est possible de renseigner des valeurs numériques. Pour cela, la démarche à suivre serait de :

- Mener une étude de terrain afin de déterminer les caractéristiques et limites associées à ce dernier
- Comparer cette étude aux demandes et exigences du système afin de proposer les solutions les plus rentables et efficaces possibles
- Adapter ensuite les dimensions du robot, du système de ramassage, les différentes mobilités possibles ainsi que leurs valeurs (vitesse, accélération…)

Annexe 4 : Complément sur la reconnaissance interne des déchets

Caractérisation des polymères grâce à la spectroscopie infrarouge, UV-Visible, Raman et RMN

Les déchets plastiques sont souvent de la catégorie des thermoplastique, ils contiennent donc des polymères linéaire ou ramifiées

	Polymères linéaires ou ramifiés	*Polymères réticulés*
Information recherchée	Identification et quantification des groupements chimiques, structure des polymères, cinétiques, mécanismes de polymérisation.	
Techniques spectroscopiques	• Raman • Infrarouge (IRTF) • RMN liquide ou solide (^{1}H, ^{13}C, ^{19}F, ^{29}Si)	*Avant le point de gel* • Raman • IRTF • RMN liquide (^{1}H, ^{13}C, ^{19}F, ^{29}Si) *Après le point de gel* • Raman • IRTF • RMN solide (^{1}H, ^{13}C, ^{19}F, ^{29}Si)

Méthodes spectroscopiques couramment utilisées dans la caractérisation des polymères

Annexe 5 : Étude du système d'absorption du filet du Jellyfishbot

Les filets absorbent les hydrocarbures en les piégeant dans leurs fibres. Le matériau des filets étant hydrophobes (polypropylène par exemple), ils n'absorbent pas l'eau donc uniquement les liquides en surface ce qui permet en plus de faire flotter le filet.

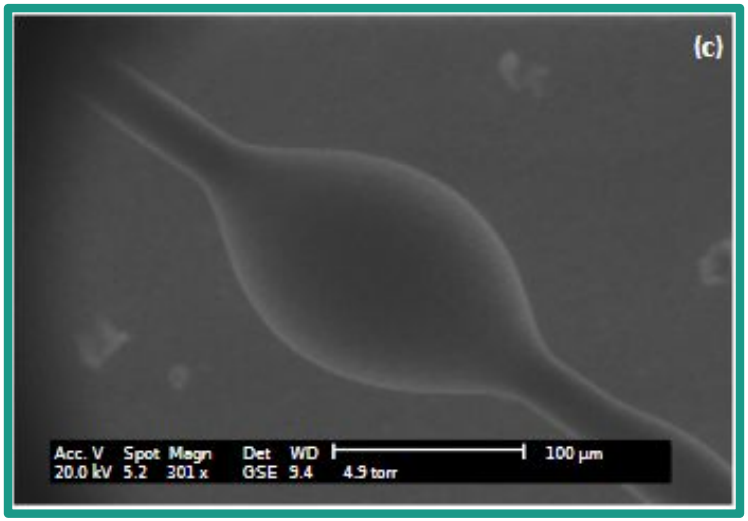


Figure 5

ESEM images of oil droplet on fibres. a) light oil; b) medium oil; c) heavy oil.

Source : https://www.researchgate.net/publication/245277749_Esem_Study_of_Oil_Wetting_Behaviour_of_Polypropylene_Fibres

Annexe 6 : Utilisation des différents capteurs dans la détection d'obstacles

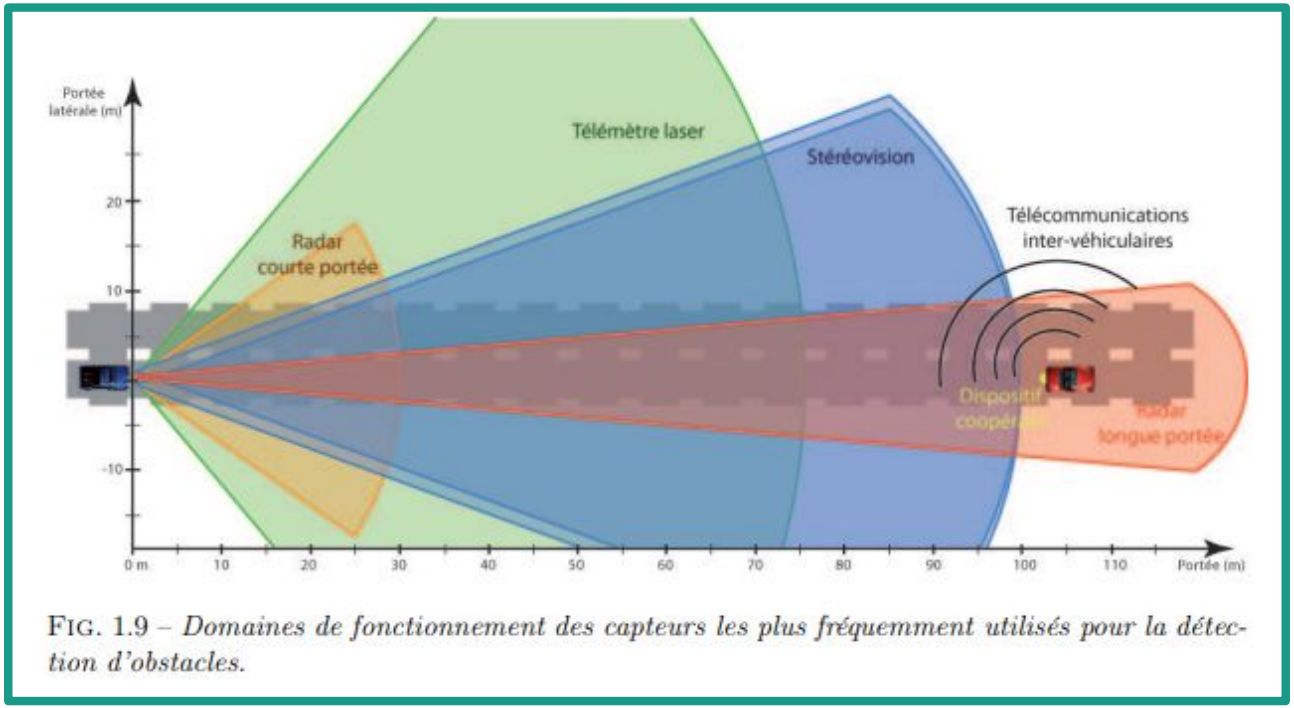


FIG. 1.9 – *Domaines de fonctionnement des capteurs les plus fréquemment utilisés pour la détection d'obstacles.*

Annexe 7 : Impact de la concentration en surface des hydrocarbures sur la flottaison du système

Puisque les hydrocarbures sont en surface, ils ont une densité plus faible que celle de l' eau :

$$d_{hydrocarbure} < d_{eau}$$

Ceci implique donc immédiatement que :

$$\mu_{hydrocarbure} < \mu_{eau}$$

Et donc par définition de la norme de la poussée d'Archimède,

$$\Pi_{hydrocarbure} < \Pi_{eau}$$

Il faut donc prendre en compte la masse volumique des hydrocarbures présents afin d'adapter la flottaison du système qui risque de couler s'il se trouve sur une nappe d'hydrocarbures en surface

Annexe 8 : Impact direct du vent sur le système

On considère que la vitesse du vent est uniforme et s'applique perpendiculairement sur le système, ce qui donne après une étude mécanique que la vitesse du vent maximale envisageable avant un basculement est d'environ :

$$v_{\max} = \sqrt{\frac{2mh}{\rho S C_x L}} = 120 km/h$$

On voit donc bien que la sécurité du système dépend beaucoup des conditions météorologiques puisque dans le cas de tempête, surtout en bord de mer, le vent peut souvent souffler à cette vitesse voire plus.

Annexe 9 : Fonction de filtrage pour l' étude des capteurs

```
def filtre(liste):
    moy = np.mean(liste)
    ecart_type = np.std(liste)
    for i in liste:
        if i < moy - 6*ecart_type or i > moy + 6*ecart_type:
            liste.remove(i)
    return(liste)
```

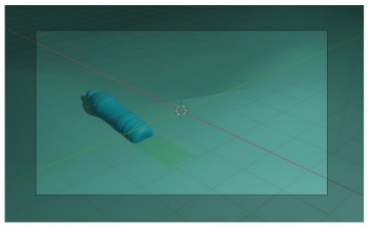
Le but de cette fonction est d'éliminer les valeurs aberrantes qui ne rentrent pas en compte dans l' étude du capteur.

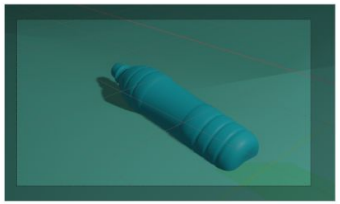
Annexe 10 : Modélisation Blender et autres méthodes de détection des déchets

Stratégie : Utiliser une détection de mouvement couplé à une reconnaissance de déchets plastiques :

I: Modelisation

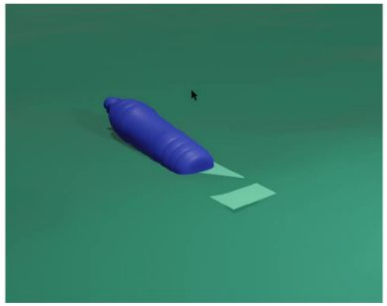
-distances différentes



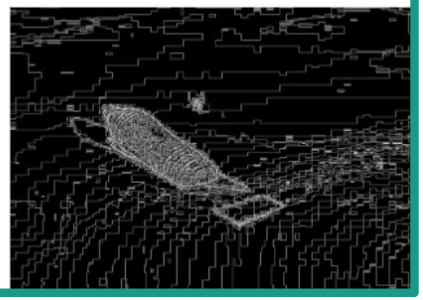


II.Détection

1.détection de mouvement : Laplacien



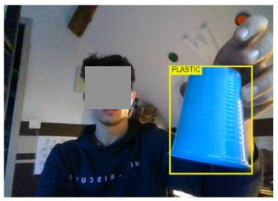
```python
import cv2

# Load the image in greyscale
img = cv2.imread('image.jpg',0)

# Apply Laplacian operator
laplacian = cv2.Laplacian(img,cv2.CV_64F)
cv2.imshow('lap',laplacian)
cv2.waitKey(0)
```



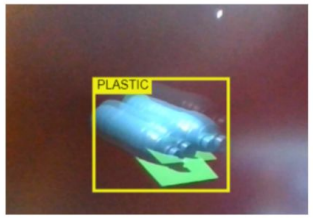
→ Le plus précis

Annexe roboflow:

2.Autre méthode: par classification



roboflow



garbage-classification-3/2 53.4% precision

Training Set 82%

14k images

-Vidéo à vitesse ralentie

Annexe 11 : Microplastiques dans le sang

Le destin de ces microplastiques qui voyagent dans notre sang reste en grande partie inconnu. Il pourrait atteindre les organes et s'y accumuler, mais les conséquences de leur présence sont actuellement inconnues. Les microplastiques semblent contaminer le sang humain *via* les muqueuses, par inhalation ou ingestion.

La moitié des échantillons contenaient du PET – *polyéthylène téréphtalate* –, que l'on retrouve dans la composition des bouteilles en plastiques ou des vêtements en polyester notamment. Plus d'un tiers des prélèvements renfermait aussi du polystyrène, utilisé entre autres pour des emballages alimentaires. Et il y avait aussi des traces de polyéthylène, que l'on retrouve dans les sacs plastiques. Ce sont des particules particulièrement fines,

https://www.futura-sciences.com/sante/actualites/corps-humain-microplastiques-identifies-sang-humain-premiere-fois-97640/

https://www.francetvinfo.fr/replay-radio/le-billet-vert/sante-pour-la-premiere-fois-une-etude-fait-etat-de-microplastiques-dans-du-sang-humain_5007026.html

Annexe 12 : Code pour l' étude des capteurs

```
def test(L):
    liste = filtre(L)
    moy = np.mean(liste)
    ecart_type = np.std(liste)
    print("moyenne =", moy)
    print("écart_type =", ecart_type)
    plt.hist(liste,bins=11) #on trace l'histogramme des valeurs mesurées
    f = lambda x : 1/(sqrt(2*pi*pow(ecart_type,2))) * exp(-pow((x-moy),2)/(2*pow(ecart_type,2)))*len(liste)*sqrt(ecart_type)
    x2 = np.linspace(520,545,len(L))
    y2 = [f(x) for x in x2]
    plt.plot(x2,y2)
    plt.show()
```

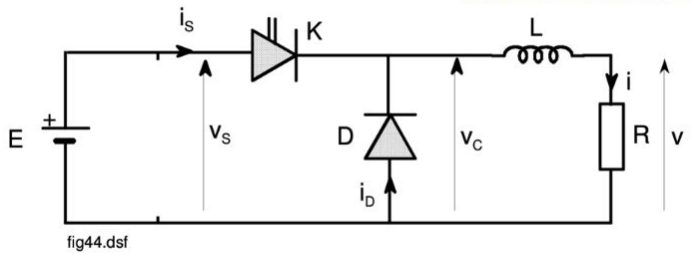
Annexe 13 : Code entier suivre l' objet le plus proche

```python
import cv2
import numpy as np
import time
from pyfirmata import Arduino, SERVO

carte = Arduino("COM5")
pin = 7
carte.digital[7].mode = SERVO
carte.digital[7].write(0)
L=[]

liste_temps = []

direct = cv2.VideoCapture(0)
direct.set(3, 640)  # largeur
direct.set(4, 480)  # hauteur

visage_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_alt.xml")

while(True):
    t0 = time.time() #on récupère le temps avant la détection
    test, image = direct.read() #direct.read() est un tupple composé d'un test puis de l'image
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) #on transforme l'image en niveau de gris
    visages = visage_cascade.detectMultiScale(gray, 1.3, 5) #on effectue la détection de visage
    for (x, y, largeur, hauteur) in visages:
        cv2.rectangle(image, (x, y), (x+largeur, y+hauteur), (255, 255, 0), 2) #on trace un rectangle autour du visage détecté
    cv2.imshow('image', image) #on affiche le résultat de l'algorithme
    (dim_x,dim_y,c)=image.shape # on récupère la dimension de l'image
    center_x=dim_x//2 # on travaille sur le centre de l'axe horizontal
    for i in range(len(visages)):
        L.append(abs((visages[i][0]+largeur)/2-center_x))
    if len(L)!= 0: #on sélectionne l'objet le plus proche du centre
        m = min(L)
        j = L.index(m) #on stocke son indice et on travaille dessus
        if (visages[j][0]+largeur)/2-center_x>10: #si l'objet se trouve à droite du centre
            print('right',(visages[j][0]+largeur)/2-center_x)
            carte.digital[pin].write(135) #on modifie la direction
        elif (visages[j][0]+largeur)/2-center_x<-10: #si l'objet se trouve à gauche du centre
            print('left', (visages[j][0]+largeur)/2-center_x)
            carte.digital[pin].write(45)  #on modifie la direction
        else: #si l'objet n'est pas éloigné du centre
            print('forward')
    del L[::] #on vide la liste des objets détectés pour éviter leur accumulation
    k = cv2.waitKey(1)
    liste_temps.append(time.time()-t0) #on stocke dans une liste le temps de la détection
    if k == ord('q') or k == 27:
        break

print(liste_temps)
print('moyenne du temps de détection =', np.mean(liste_temps))
direct.release()
cv2.destroyAllWindows()
```

Annexe 14 : Hacheur et moteurs

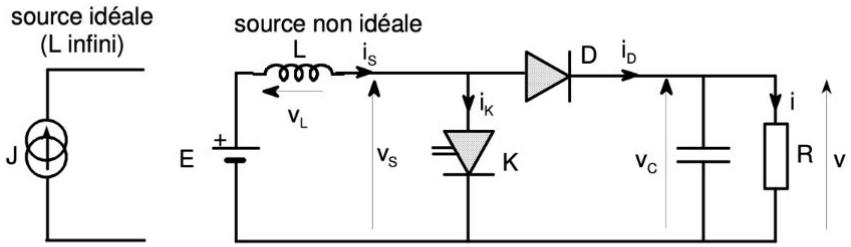
Afin de réaliser un asservissement en vitesse des moteurs, on peut utiliser les différents hacheurs.

HACHEUR SERIE (BUCK)



Hacheur série

HACHEUR PARALLELE (BOOST)



hacheur parallèle (source de courant non idéale)

HACHEUR INDUCTIF (BUCK-BOOST)

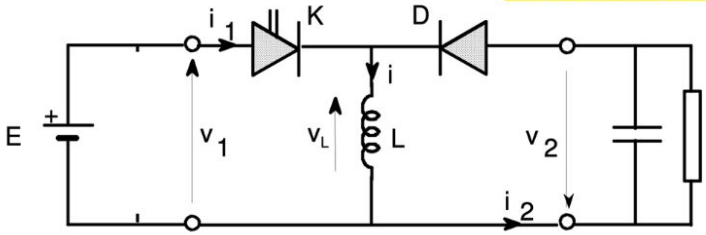
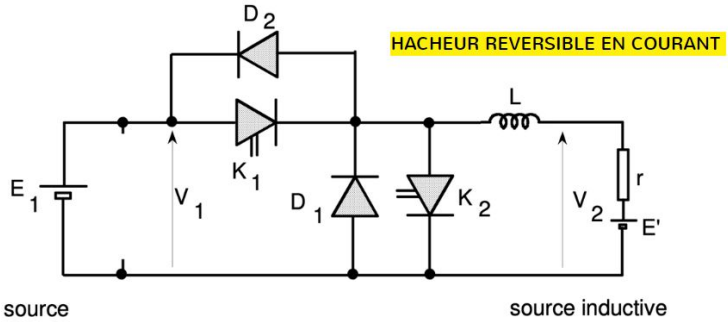


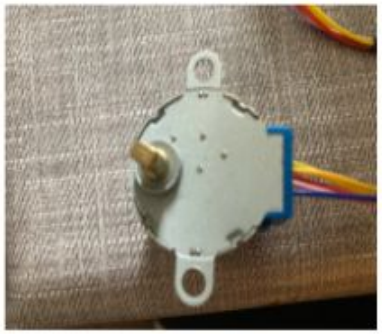
Schéma de principe du hacheur BUCK-BOOST

HACHEUR REVERSIBLE EN COURANT



Annexe 15 : Deux moteurs pour l' asservissement en direction

Moteur pas à pas 28BYJ-48:



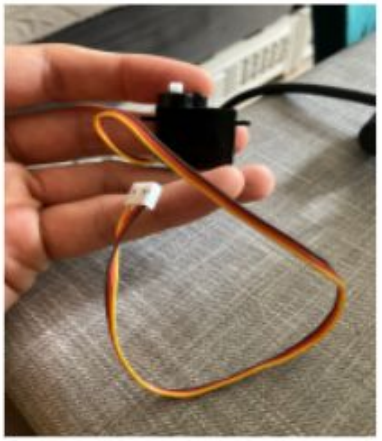
-**vitesse de rotation basse** :
2,64 s pour 180° (moyenne sur 5 mesures:2.6s,2.4s,2.6s,2.8s,2,5s)

-rotation sur 360 °

-précis : pas de 0,7 °

-Couple suffisant (2N.cm)

Servomoteur Grove:



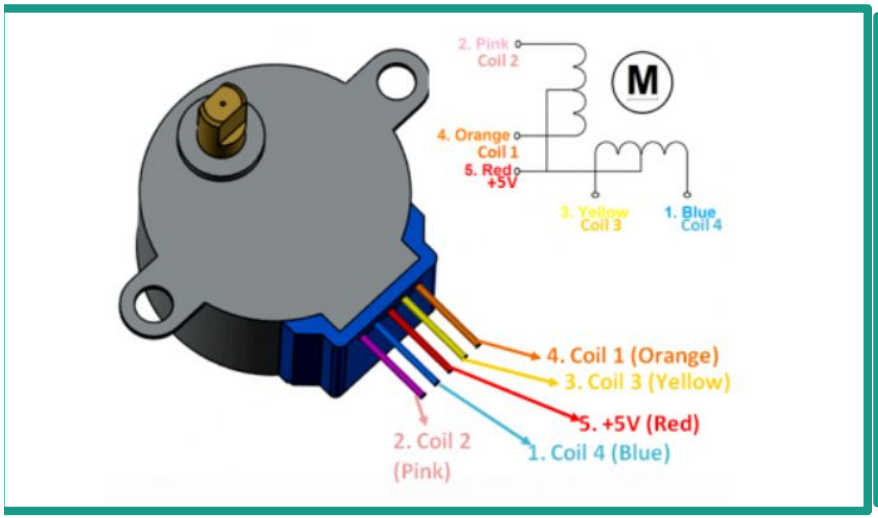
-peut uniquement faire du 0 à 180°

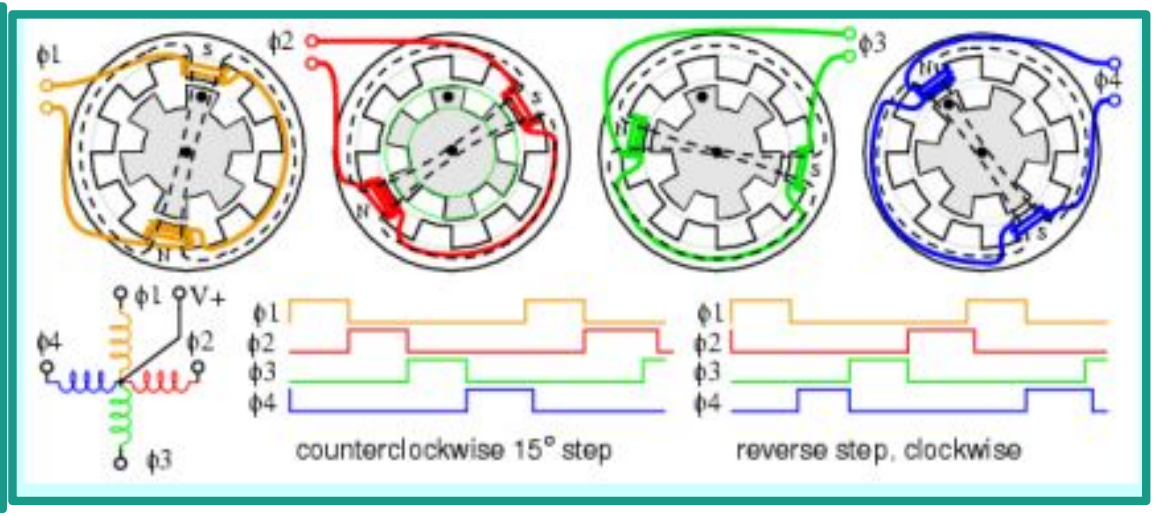
-**vitesse de rotation élevée** : environ une demi seconde pour faire 180° (0,53 , 0,57s)

-moins précis :précision d'environ 1,5°

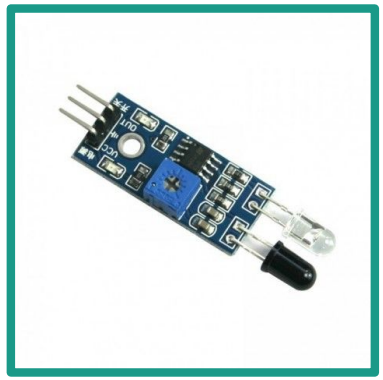
-couple suffisant

Annexe 16 : Moteur pas à pas

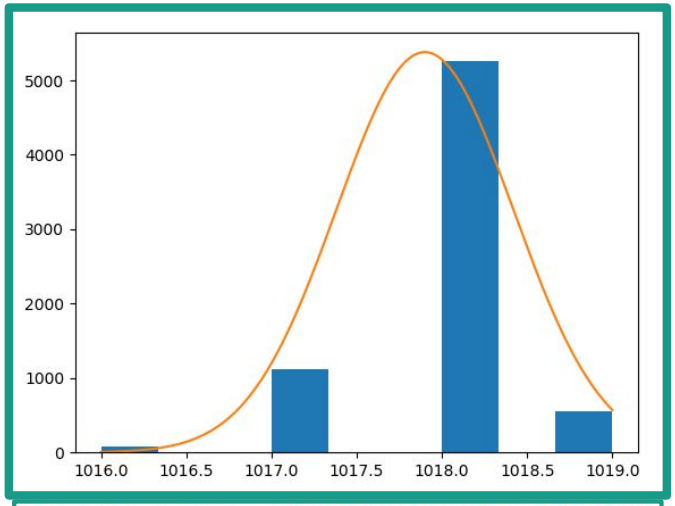




Annexe 17 : Étude du capteur infrarouge



```
import numpy as np
import matplotlib.pyplot as plt
from math import *
def test(n=10):
    liste = [1017,1018,1018,1018,1017,1017,1018,1016,1018,1018,1019,1018,1018,1018,1018
    moy = np.mean(liste)
    ecart_type=np.std(liste)
    print('moyenne =',moy)
    print('écart-type=',ecart_type)
    plt.hist(liste,bins=9)
    sigma = ecart_type
    f = lambda x : 1/(sqrt(2*pi*pow(sigma,2))) * exp(-pow((x-moy),2)/(2*pow(sigma,2)))
    x2 = np.linspace(1015,1019,3900)
    y2 = [len(liste) * f(x) for x in x2]
    plt.plot(x2,y2)
    plt.show()
    return()

test()
```




```
écart-type= 0.5190456597458302
```

Annexe 18 : Pompe centrifuge

La puissance hydraulique fournie par la pompe est donnée par la relation :

$$P_{hydraulique} = \rho\, g\, Q\, h$$

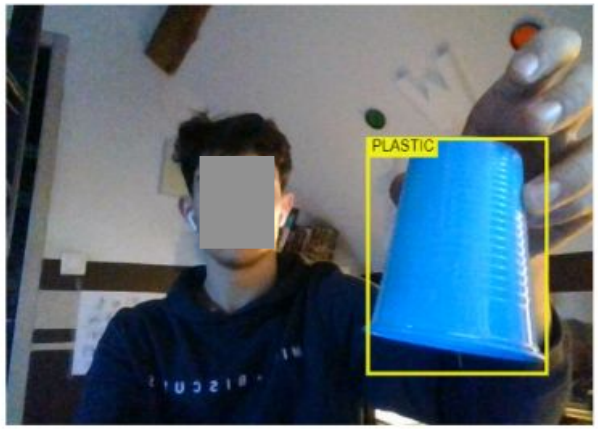
Dans laquelle :

- $P_{hydraulique}$ est exprimée en watts
- ρ est la masse volumique du liquide (kg/m^3)
- g est l'accélération de pesanteur soit 9,81 m/s^2
- Q est le débit volumique du liquide exprimé en m^3/s
- h est la hauteur manométrique de la pompe exprimée en mètres de colonne d'eau

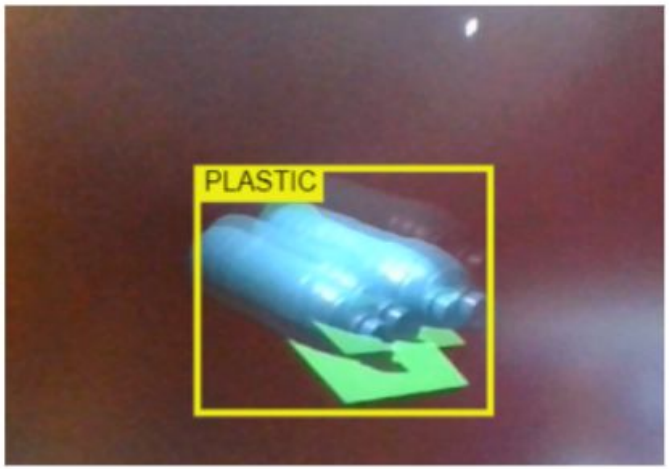
La hauteur manométrique est la hauteur d'une colonne de liquide qui déterminerait une pression statique égale à la pression de refoulement.

Risques de cavitation

Annexe 19 : Utilisation d'un classifier pour l' entraînement d'une intelligence artificielle



roboflow



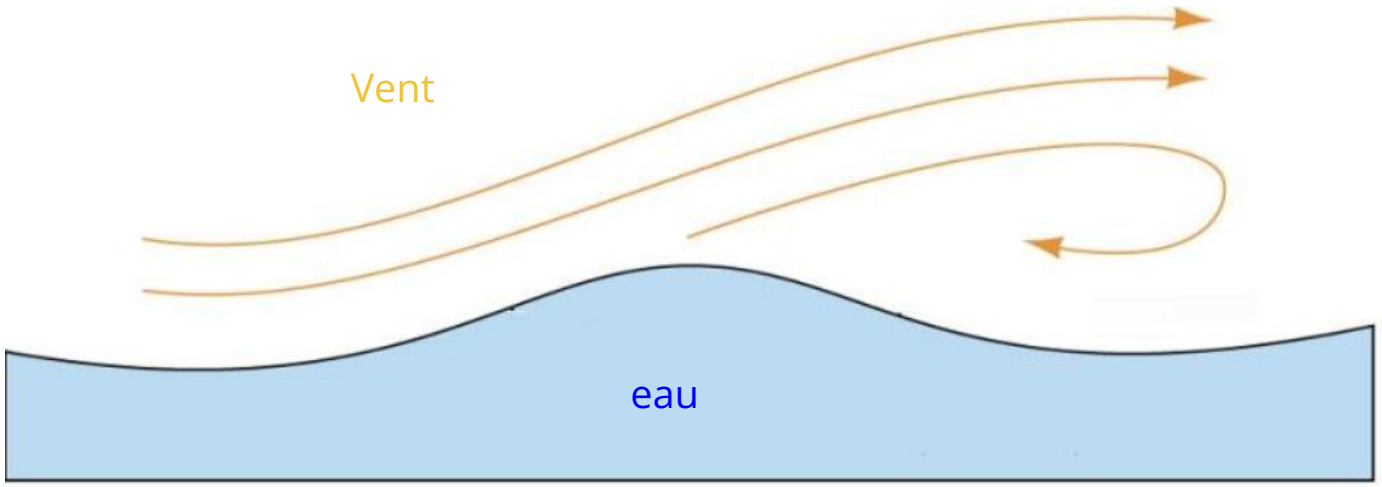
garbage-classification-3/2 53.4% precision

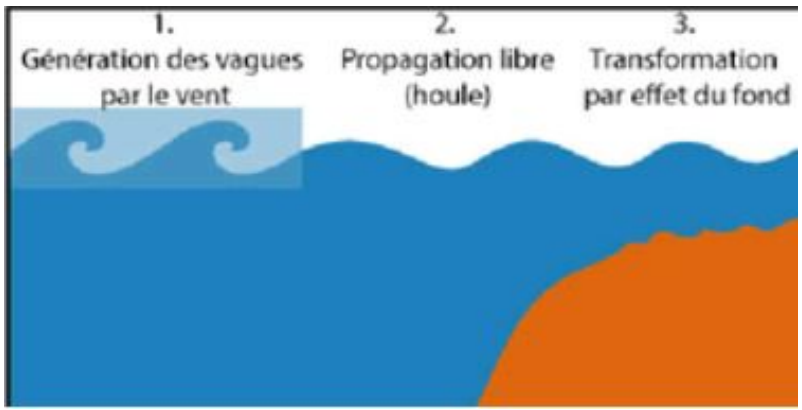
Training Set 82%

14k images

-Vidéo à vitesse ralentie

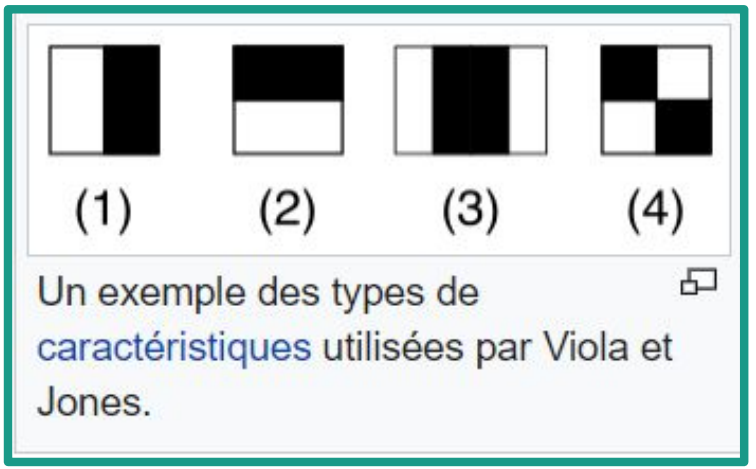
Annexe 20 : Naissance de la houle





Annexe 21 : Méthode de la cascade de Haar

Méthode de Viola et Jones : le principe consiste à étudier les caractéristiques d'une image en fonction de l'objet que l'on souhaite détecter au lieu d'étudier chaque pixel



Un exemple des types de caractéristiques utilisées par Viola et Jones.

Annexe 22 : Recyclage