

MICRO-315 Miniprojet

Les aventures incroyables de Jean-Bot Baptiste I

Le raccourceur des circuits

Un projet de robotique de Julian Donevsky et Olof Lissmats

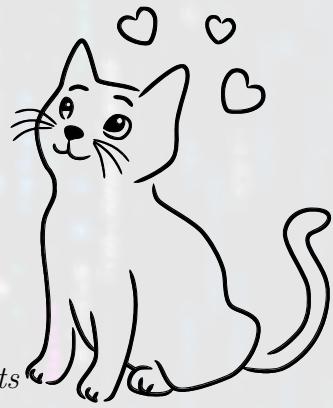


Table des matières

1	Introduction	1
2	Méthodologie	1
2.1	Schéma général du programme	1
2.2	La fonction extract_color_bands	2
3	Problèmes apparus lors du projet	2
4	Points à améliorer	2
4.1	Créer une thread propre pour la détection d'obstacles	3
4.2	Rendre la détection de couleurs plus fiable	3

1 Introduction

Dans le cadre de ce projet, nous avons créé un programme qui permet au robot de danser sur une musique en particulier en fonction d'un drapeau de pays que l'utilisateur lui présente. Le robot utilise ses capteurs de proximités infrarouges pour s'assurer qu'aucun obstacle ne bloque son chemin lorsqu'il danse. Si c'est le cas, il arrête temporairement sa danse jusqu'à ce que l'obstacle soit dégagé.

2 Méthodologie

2.1 Schéma général du programme

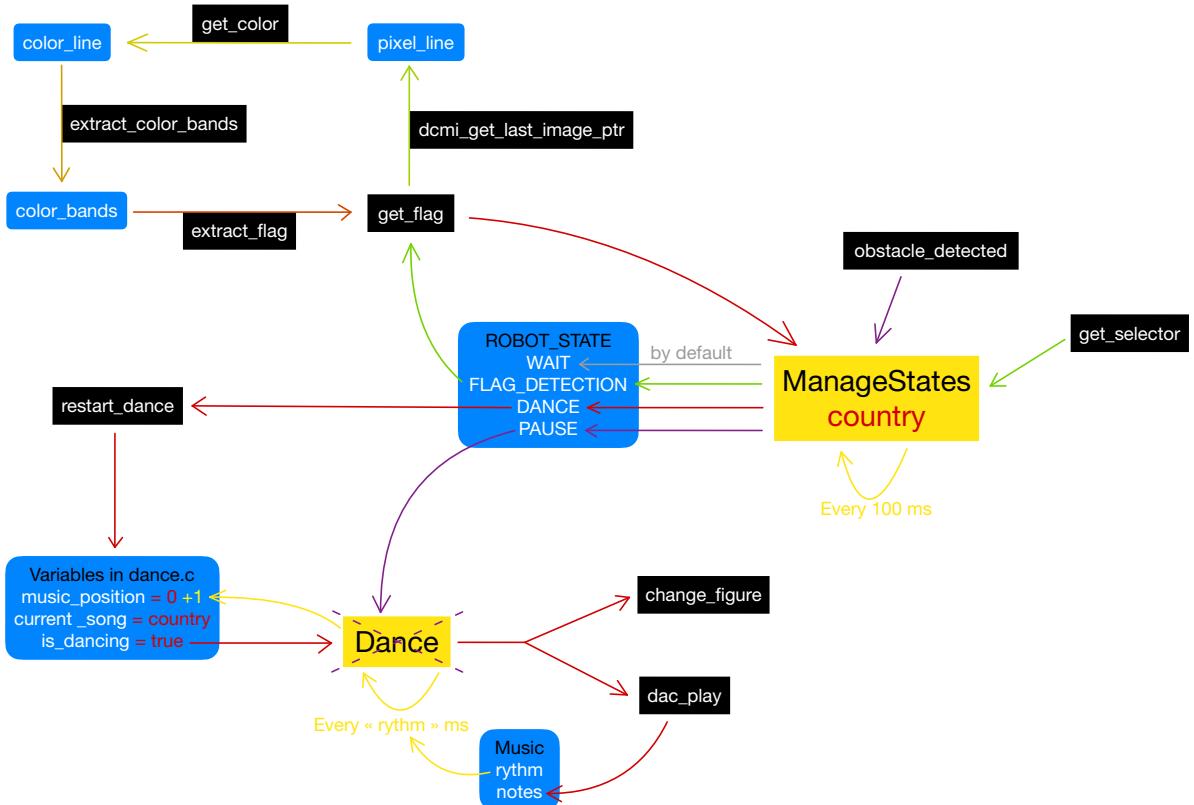


FIGURE 1 – Schéma général du programme. En jaune les threads, en noir les fonctions et en bleu (rouge dans le cas de country) les variables utiles au bon déroulement du programme.

L'utilisateur tourne le selecteur (peu importe de combien de cran). La détection se fait en appellant la fonction `get_selector` dans le thread **ManageStates**, qui met ensuite l'état du robot à **FLAG_DETECTION** ce qui a pour conséquence d'appeler la fonction `get_flag` (flèche en vert).

A chaque appel de `get_flag`, une image sur la caméra est capturée et on stocke une ligne de pixel dans la variable `pixel_line`. On convertit chaque pixel de `pixel_line` en un pixel de couleur (blanc, rouge, vert, bleu, indéfini) grâce à la fonction `get_color` qu'on stocke ensuite dans `color_line`. La fonction `extract_color_bands` s'occupe d'analyser `color_line` (voir section 2.2) pour en extraire les grandes bandes de même couleur correspondant au drapeau présenté. Finalement, `extract_flag` en déduit le drapeau et le retourne à `get_flag`.

Si la fonction `get_flag` n'a pas réussi à trouver le drapeau, elle continue de s'exécuter en boucle jusqu'à obtenir un drapeau connu pendant au maximum 10 secondes. Passé ce délai, le robot retourne dans l'état **WAIT**. Lorsqu'un drapeau connu est enfin trouvé, le thread **ManageStates** appelle la fonction `restart_dance` qui réveille le thread **Dance** (flèche en rouge). En effet, la variable `is_dancing` est initialisée à `false` et inhibe le thread en le faisant simplement dormir dans une boucle de 100 ms. Lorsqu'elle passe à `true`, celui-ci sort de son sommeil et commence à jouer la musique `current_song` (remise au pays du drapeau détecté par `restart_dance`). A chaque note jouée, les fonctions `change_figure` et `dac_play`

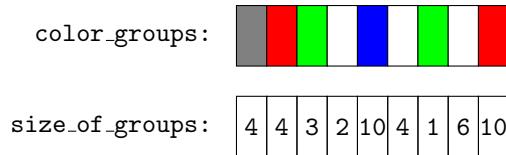


sont appelée pour changer l'état des motors (de façon aléatoire) et jouer le son voulu, respectivement. De plus, la variable `music_position` est incrémentée, celle-ci indiquant quelle note et rythme jouer en pointant sur la musique en cours.

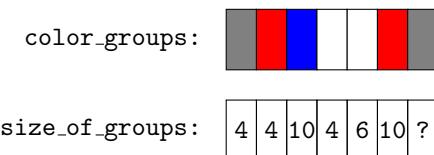
2.2 La fonction extract_color_bands



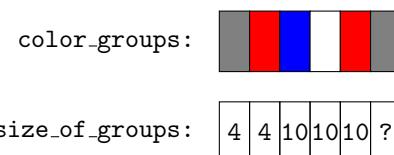
(a) Une fois les valeurs RGB de chaque pixel sont convertis à des couleurs, elles sont stockées de cette manière. La couleur grise représente une couleur non définie. La caméra fournit en fait une ligne de 640 pixels, mais dans cet exemple seulement 40 sont représentés.



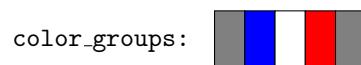
(b) Les pixels sont groupés en fonction de leurs couleurs dans les deux arrays `color_groups` et `size_of_groups` qui contiennent la couleur et la taille des groupes respectivement.



(c) Les groupes avec une taille supérieur à une seuile spécifique, ici 3, sont placés au début des arrays, c'est à dire que les groupes trop petits sont supprimés. Cela se fait pour ammortir l'effet de bruit.



(d) Les groupes consécutives ayant la même couleur sont fusionnés, et les groupes suivants sont décalés d'un placement.



(e) Les groupes d'une taille inférieure d'une seuile spécifique, ici 8, sont fusionnés avec les groupes non-définis entourants.

FIGURE 2 – Un schéma qui explique comment fonctionne la détection des drapeaux.

3 Problèmes apparus lors du projet

4 Points à améliorer

Il y a maintes points à améliorer afin d'augmenter la qualité du robot, dont les plus importants sont présentés ci-dessous.



4.1 Créez une thread propre pour la détection d'obstacles

En ce moment le robot vérifie l'absence d'obstacles en appellant une fonction au début de la thread qui gère les inputs et les états. Par conséquent, un obstacle peut se présenter après cet appel, lorsque le robot est occupé par d'autres tâches, sans causer un arrêt d'activité. Cela se produit par exemple pendant la détection des drapeaux où le robot est bloqué dans une boucle.

L'ajout d'une thread complètement dédiée à la détection d'obstacle permettrait d'y réagir à chaque instant. En revanche cela impliquerait aussi une nécessité de faire communiquer cette thread avec les autres, c'est à dire introduire plus de variables globales. La raison pour laquelle cela n'est pas encore fait est une manque de temps et la reorganisation du code nécessaire.

4.2 Rendre la détection de couleurs plus fiable

La caméra du robot fournit une image de 640 fois 480 pixels sous le format RGB565. Cela signifie que l'intensité de rouge, vert et bleu sont représentés sur 5, 6 et 5 pixels respectivement. Afin de déterminer la couleur représentée, les écarts entre ces trois valeurs sont analysées. Si R dépasse G et B par un seuil spécifique, c'est admis que c'est bien rouge qui est représenté. La détection de vert et bleu se fait de la même façon, et au cas de blanc, le test et fort ressemblant.

Le problème de cette démarche est que les valeurs RGB sont dépendantes de la luminosité, ce qui implique des performances dépendantes de la luminosité. Le robot pourrait donc être capable de détecter une couleur à un moment donné, or incapable quelques mètres à côté ou un certain temps après. De sorte d'y prévenir un système qui prends en compte la luminosité ambiante pourrait être mis en place. Sous supposition que les valeurs RGB sont représentées sur la même échelle, c'est à dire le même nombre de bits, le pourcentage dont une valeur dépasse les autres pourrait être analysé par exemple.

