**RAPPORT : PROJET SYSTÈMES EMBARQUÉS ET ROBOTIQUES**

Notre projet consiste à ce que le robot E-Puck2 suive des lignes de différentes couleurs et évite les possibles obstacles sur sa trajectoire.

1. **Méthodes de travailles**

Nous commençons par séparer le projet en deux taches majeures : la détection de différentes couleurs à l'aide de la caméra (PO8030D 640x480) et la détection d'obstacles grâce aux capteurs infrarouges *(et time of flight).*

* Pour différencier les couleurs, nous partons comme base pour le projet le TP4 (maintenir le robot a une distance fixe d'une ligne noire). Chaque pixel code la couleur sur 16 bits au format RGB565 (5 bits pour le rouge et le bleu, et 6 bits pour le vert), nous avons donc commencer par extraire les bits pour chaque couleur et les mettre à leur échelle respective (0 à 31 pour rouge et bleu, 0 à 63 pour vert). La remise à l'échelle permet de diminuer l'amplification du bruit, notamment dans le cas du rouge et du vert, puisque les deux échelles sont augmentées de 0 à 255.

|  |
| --- |
|  |
| Figure 1 : Détection ligne rouge sur fond noir sans remise à l'échelle (valeur entre 0 et 255)  *Note : L'abscisse horizontal représente la position horizontale des pixels sur la camera* |

|  |  |
| --- | --- |
|  |  |
| Figure 2 : Détection ligne rouge avec remise à l'échelle d'origine (0 à 31) sur fond noir sans filtrage | Figure 3 : Détection ligne rouge sur fond noir avec remise à l'échelle et avec filtrage (utilisation d'une valeur seuil pour éliminer le bruit) |

Le fond noir permet de mieux distinguer les lignes du fond, et le seuillage permet d'éliminer les reflets de lumière sur ce dernier. L'éclairage de l'environnement où se trouve l'E-Puck2 est clé pour une bonne distinction, s'il fait trop sombre, la couleur de la ligne se perdra dans le bruit noir, si l'espace est trop lumineux, tous les pixels seront saturés à 255 rendant impossible la lecture.

Pour s'adapter à l'environnement, les seuils utiliser lors de la filtration sont réglable manuellement en utilisant le sélecteur (le seuil minimum vaut 25% et le seuil maximal vaut 96% de la valeur maximal).

Nous nous sommes heurtés au problème suivant, pour une ligne de couleur distincte (bleu par exemple), les pixels rouges et verts donnés des valeurs d'intensité relativement élevée, alors que la scène était uniquement composée d'un fond noir et d'une ligne bleue.

|  |
| --- |
|  |
| Figure 4 : Valeurs d'intensité des pixels rouges pour la détection d'une ligne bleu sur fond noir. |

De plus nous avons remarqués, que lorsque nous exposions une ligne sur un fond noir, nous obtenions des valeurs d'intensité élevés voir maximales pour cette couleur pendant un court instant, puis les valeurs d'intensité chute drastiquement. Ceci est dû au "auto white balance" (AWB ou balance des blancs), c'est un réglage de la camera PO8030D qui modifie les couleurs en fonction de l'éclairage. Nous supposons que l'algorithme utilise suppose que l'image est grise et réduit l'intensité des pixels les plus colorées. L'AWB est problématique pour notre projet, car il diminue l'intensité des couleurs et rend la distinction entre couleurs plus difficile. Nous désactivons donc manuellement ce réglage et augmentons le contraste de la caméra afin d'améliorer la saturation des pixels pour pouvoir mieux distinguer les couleurs.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Rouge | Vert | Bleu | Cyan | Jaune | Magenta |
| Max Pixels R | 26 | 21 | 12 | 21 | 30 | 20 |
| Max Pixels V | 25 | 46 | 22 | 52 | 61 | 31 |
| Max Pixels B | 15 | 17 | 15 | 26 | 24 | 17 |

Tableau 1 : Valeurs maximales des pixels pour des lignes de différents couleurs sur fond noir (awb on)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Rouge | Vert | Bleu | Cyan | Jaune | Magenta |
| Max Pixels R | 31 | 20 | 14 | 20 | 30 | 30 |
| Max Pixels V | 22 | 50 | 22 | 51 | 59 | 24 |
| Max Pixels B | 19 | 18 | 22 | 30 | 22 | 24 |

Tableau 2 : Valeurs maximales des pixels pour des lignes de différents couleurs sur fond noir (awb off)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Rouge | Vert | Bleu | Cyan | Jaune | Magenta |
| Max Pixels R | 31 | 26 | 18 | 26 | 31 | 31 |
| Max Pixels V | 31 | 60 | 26 | 63 | 63 | 31 |
| Max Pixels B | 25 | 23 | 25 | 31 | 25 | 27 |

Tableau 3 : Valeurs maximales des pixels pour des lignes de différents couleurs sur fond noir (awb off + contraste augmenté de 64 à 80)

*Note : Les cases en oranges représente une saturation.*

Les lignes furent imprimées avec les imprimantes de l'EPFL en codant les codes couleurs de la facon suivante : 0xFF0000 (rouge), 0x00FF00 (vert), 0x0000FF (bleu), 0x00FFFF (cyan), 0xFFFF00 (jaune) et 0xFF00FF (magenta).

Le bleu en question (0x0000FF) ne sature pas à la lumière ambiante. Cela peut être dû au fait que les imprimantes utilises le système de couleur CMYK (cyan, magenta, jaune et noir), l'impression du bleu "pure" 0x0000FF donne un bleu plus sombre et moins intense que le bleu parfait théorique, car il est réalisé par addition des couleurs complémentaires : cyan + magenta = bleu. Une étude de couleur a été effectué avec différents bleus. Un bleu optimal fut trouvé 0x0070C0 (R = 0, G = 112 et B = 192).

|  |  |
| --- | --- |
|  | Bleu 0x0070C0 |
| Max Pixels R | 13 |
| Max Pixels V | 40 |
| Max Pixels B | 31 |

Tableau 4 : Valeurs maximales des pixels pour une ligne bleu 0x0070C0 sur fond noir (awb off + contraste augmenté de 64 à 80)

Similairement une étude a été réalisé pour trouver un vert optimal afin de baisser l'intensité des pixels rogue et garantir la saturation des pixels rouges pour gagner en robustesse. Le vert optimal trouvé fut 0x00E650 (R = 0, G = 230, B = 80). Il permet d'obtenir une saturation des pixels verts et maintenir l'intensité des pixels rouge au même niveau que précédemment.

|  |  |
| --- | --- |
|  | Vert 0x00E650 |
| Max Pixels R | 26 |
| Max Pixels V | 63 |
| Max Pixels B | 25 |

Tableau 4 : Valeurs maximales des pixels pour une ligne bleu 0x0070C0 sur fond noir (awb off + contraste augmenté de 64 à 80)

*Note : Lors de l'impression les reglages de l'imprimmante sont tres important : utiliser "Adobe Color Magement" et non "Print Color Management" pour le la gestion des couleurs.*

On constate que les valeurs de bleu sont très sensibles à l'éclairage, cette couleur étant plus sombre que les autres son intensité est donc plus faible. Pour bien détecter le bleu, il est primordial de travailler avec un bon éclairage, c’est-à-dire pas de rayons de lumières directes sur la feuille et pas trop sombre.

Par la suite, pour des raisons pratiques nous avons décidé de ramener le vert sur la même échelle que le bleu et le rouge a savoir 0 à 31, en gardant les 5 bits de poids fort.

Une autre problème corps à notre projet est la fixation d'un miroir sur le robot, de sorte à ce que la caméra capture le support sur lequel il roule et n'empêche pas l'utilisation du capteur de distance "Time of Flight" et des capteurs infrarouges IR0 et IR7.

|  |
| --- |
|  |
| Figure 7 : Photo E-Puck2 – Considérations à prendre pour fixer le miroir |

|  |  |
| --- | --- |
|  |  |
| Figure 8 : E-Puck2 avec miroir fixé | |

N'ayant pas d'emplacement pour fixer le miroir proche de la caméra, la solution retenue fut de coller un morceau de carton avec une forme désirée sur le robot. Cette solution a l'avantage d'être stable dans le temps : le miroir reste fixe durant le déplacement du robot, garantissant le même angle de vue à tout moment. Cependant, les capteurs Time of Flight, IR0 et IR7 ne sont plus utilisable pour notre projet, le support en carton rendant les mesures impossibles.

L'éclairage de la scène est très important, comme le fond noir et les lignes de couleurs sont créés par impression, une source de lumière place au-dessus du robot pour éclairer la scène peut créer de reflets indésirables sur le noir et sur la couleur de la ligne (rouge, vert et bleu), augmentant l'intensité et pouvant mener à une saturation non désirée. Il est donc préférable d'éviter d'illuminer la scène depuis le plafond mais privilégier d'avantage la lumière ambiante (ex. bibliothèque RLC durant la journée). En règle générale, il faut éviter que la lumière tombe directement sur la feuille (augmentation des valeurs), mieux vaut travailler à l'ombre.

* Détection d'obstacles grâce aux infrarouges. Comme vu précédemment les capteurs IR0 et IR7 étant indisponible pour notre application, nous utiliseront les capteurs IR restant à savoir :

|  |
| --- |
|  |
| Figure 9 : Position des capteurs IR |

//attention ici il faut inverser front et back//Pour pouvoir détecter les obstacles devant le robot, nous inversons l'orientation du robot, les capteurs IR3 et IR4 deviennent les capteurs avant, afin qu'un obstacle ne percute pas le montage du miroir. Ainsi, le sens de déplacement est inverse et la caméra se retrouve à l'arrière du robot. De plus nous inversons l'image avec un réglage de la caméra.

1. **Analyses**
2. **Conception**
3. **Analyse**

*Seulement ce qui n’est pas dans le code, comme structure générale, tests initiaux, graphes de capteurs, etc Respectez le format (sections numérotées, figures et tables numérotées avec légendes, axes sur les graphes, références)*

*Clarté et propreté du rapport (Numéros de sections, légendes aux figures, axes aux graphiques, citations et références)*