avril 2020

**ANNEXE 1: DONNEES POUR LE CALAGE DE LA CAMERA DU TRUGGY 2020**

1. **INTRODUCTION :**

Suite à la TRR 2019 (voir résultats de la course en annexe de ce document), l’équipe des COOLs a choisi de développer un TRUGGY avec guidage par caméra embarquée, cette solution ayant fait preuve de la meilleure performance possible pour cette course « Roulants », jusqu’à ce jour.

1. **CHOIX TECHNIQUES :**

* le modèle réduit TRUGGY, robot 4 x 4, utilisé en 2019 est conservé
* la caméra étudiée en 2019, modèle Pi Camera Module V2.1, est actuellement en test
* le calculateur de bord est un JETSON NANO ou un Raspberry PI4. Il enverra les consignes de pilotage (vitesse et angles de braquage) à un ARDUINO MEGA, qui commandera l’ESC+ son moteur, le servo de direction, et qui enverra l’odométrie au calculateur
* La carte JETSON ou Raspberry sera complétée par un module WIFI communiquant avec un ordinateur portable qui pourra assurer la commande et le contrôle du véhicule pendant les essais de développement.
* l’utilisation d’un gyroscope pourra être une option (à étudier).

1. **LA CAMERA ‘ PI Camera Module V2.1’:**

Dimensions 25x24x9 mm, masse 3g, 8 Megapixels

Plusieurs modes vidéo possibles: 1080p30, 720p60, 640x480p60/90, …

Résolution du capteur : 3280 x2464 pixels, de taille carrée 1,12 µ

En conséquence une image sur le capteur de 3,68 x 2,76 mm

Objectif : 6,3 mm, focale 3,04 mm

Champ de vue : 62,2 ° en horizontal et 48,8 ° en vertical

Jusqu’à 90 images/s max (mode video ?) ?

Rapport Signal/Bruit : 36 dB

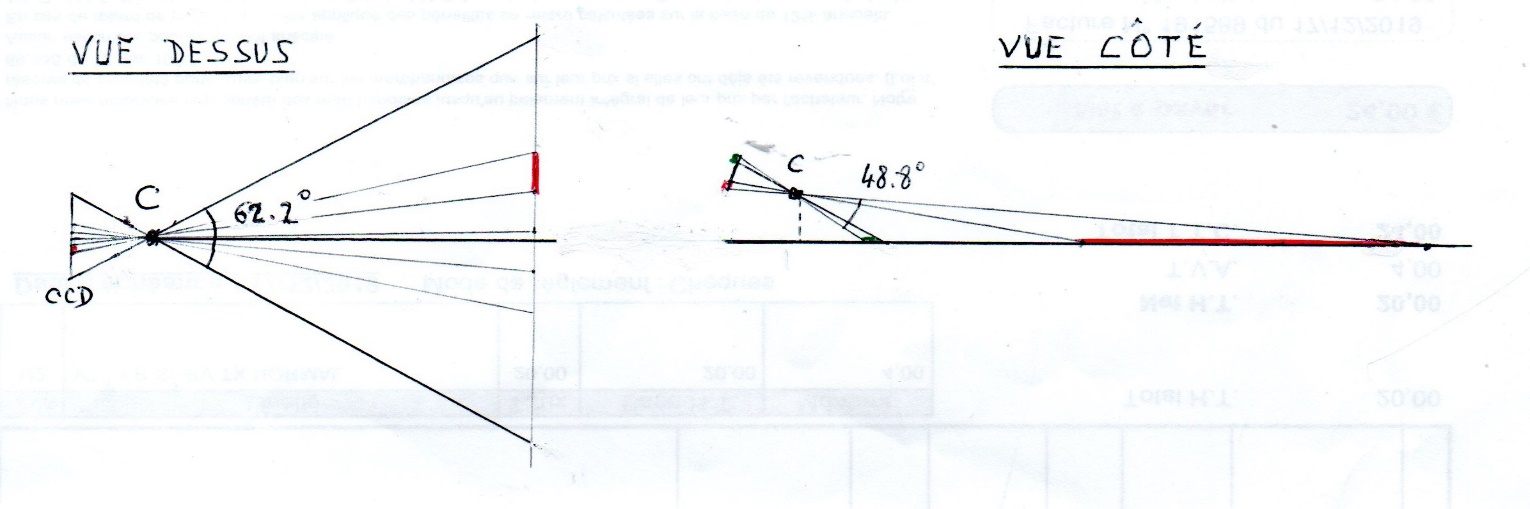
Format images : JPEG, GIF, …

Plusieurs modes d’exposition (auto, nuit, spotlight, neige, …)

Autres données : cf notice technique constructeur

**Utilisation sur le TRUGGY :**

Considérations d’optique géométrique pour le positionnement de la caméra :

La caméra pointe la bande blanche sur le sol de la piste de la TRR en étant inclinée d’un certain angle. La figure ci-dessus représente les champs de vue (données constructeur) de la caméra.

Pour guider le Truggy sur la piste, on doit détecter et suivre la bande blanche le plus en avant possible pour anticiper la direction que doit suivre le Truggy.

Les tests effectués en 2019 par Cédric ont défini une méthode de traitement de l’image permettant de détecter un « SAUT » de flux lumineux lors du passage de la bande noire de la piste à la bande blanche.

La caméra devra donc être positionnée et orientée de manière à détecter ce « SAUT » noir/blanc le plus en avant possible.

1. **CHOIX DES PARAMETRES IMAGES ET TRAITEMENT :**

**4.1 Format de l’image :**

Sur la caméra V2, plusieurs formats sont utilisables, certains avec binning. Le tableau ci -après, non exhaustif, en définit quelques uns :

| **#** | **Resolution** | **Aspect Ratio** | **Framerates** | **Video** | **Image** | **FoV** | **Binning** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1920x1080 | 16:9 | 0.1-30fps | x |  | Partial | None |
| 2 | 3280x2464 | 4:3 | 0.1-15fps | x | x | Full | None |
| 3 | 3280x2464 | 4:3 | 0.1-15fps | x | x | Full | None |
| 4 | 1640x1232 | 4:3 | 0.1-40fps | x |  | Full | 2x2 |
| 5 | 1640x922 | 16:9 | 0.1-40fps | x |  | Full | 2x2 |
| 6 | 1280x720 | 16:9 | 40-90fps | x |  | Partial | 2x2 |
| 7 | 640x480 | 4:3 | 40-90fps | x |  | Partial | 2x2 |

Modes with full field of view (FoV) capture from the whole area of the camera’s sensor (2592x1944 pixels for the V1 camera, 3280x2464 for the V2 camera). Modes with partial FoV capture from the center of the sensor. The combination of FoV limiting, and [binning](http://www.andor.com/learning-academy/ccd-binning-what-does-binning-mean) is used to achieve the requested resolution.

The various FoVs for the V2 camera are illustrated in the following image:

[](https://picamera.readthedocs.io/en/release-1.12/_images/sensor_area_2.png)

The input mode can be manually specified with the *sensor\_mode* parameter in the **[PiCamera](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera" \o "picamera.PiCamera)** constructor (using one of the values from the # column in the tables above). This defaults to 0 indicating that the mode should be selected automatically based on the requested **[resolution](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.resolution" \o "picamera.PiCamera.resolution)** and **[framerate](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.framerate" \o "picamera.PiCamera.framerate)**. The rules governing which input mode is selected are as follows:

* The mode must be acceptable. Video modes can be used for video recording, or for image captures from the video port (i.e. when *use\_video\_port* is True in calls to the various capture methods). Image captures when *use\_video\_port* is False must use an image mode (of which only two exist, both with the maximum resolution).
* The closer the requested **[resolution](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.resolution" \o "picamera.PiCamera.resolution)** is to the mode’s resolution the better, but downscaling from a higher input resolution is preferable to upscaling from a lower input resolution.
* The requested **[framerate](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.framerate" \o "picamera.PiCamera.framerate)** should be within the range of the input mode.
* The closer the aspect ratio of the requested **[resolution](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.resolution" \o "picamera.PiCamera.resolution)** to the mode’s resolution, the better. Attempts to set resolutions with aspect ratios other than 4:3 or 16:9 (which are the only ratios directly supported by the modes in the table above) will choose the mode which maximizes the resulting FoV.

( A few examples are given below to clarify the operation of this heuristic (note these examples assume the **V1 camera module**):If you set the **[resolution](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.resolution" \o "picamera.PiCamera.resolution)** to 1024x768 (a 4:3 aspect ratio), and **[framerate](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.framerate" \o "picamera.PiCamera.framerate)** to anything less than 42fps, the 1296x972 mode will be selected, and the camera will downscale the result to 1024x768. If you set the **[resolution](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.resolution" \o "picamera.PiCamera.resolution)** to 1280x720 (a 16:9 wide-screen aspect ratio), and **[framerate](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.framerate" \o "picamera.PiCamera.framerate)** to anything less than 49fps, the 1296x730 mode will be selected and downscaled appropriately.

Setting **[resolution](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.resolution" \o "picamera.PiCamera.resolution)** to 1920x1080 and **[framerate](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.framerate" \o "picamera.PiCamera.framerate)** to 30fps exceeds the resolution of both the 1296x730 and 1296x972 modes (i.e. they would require upscaling), so the 1920x1080 mode is selected instead, although it has a reduced FoV.

A **[resolution](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.resolution" \o "picamera.PiCamera.resolution)** of 800x600 and a **[framerate](https://picamera.readthedocs.io/en/release-1.12/api_camera.html" \l "picamera.PiCamera.framerate" \o "picamera.PiCamera.framerate)** of 60fps will select the 640x480 60fps mode, even though it requires upscaling because the algorithm considers the framerate to take precedence in this case.

Any attempt to capture an image without using the video port will (temporarily) select the 2592x1944 mode while the capture is performed (this is what causes the flicker you sometimes see when a preview is running while a still image is captured).

* 1. **Format choisi pour les premiers essais:**

Le format 320 x 240 pixels, soit 76800 pixels à analyser (pour 8 000 000 pixels disponibles sur le CCD de la caméra !) a été choisi dans un premier temps, ce format est une option du logiciel OPEN CV.

**Il correspond au format de base/natif 3280 x 2464 du tableau, mais a été fortement binné, un facteur 10.**

* 1. **Traitement de l’image :**

L’image initiale est en couleurs. Elle est donc d’abord transformée par OPEN CV en niveaux de gris dans un premier temps.

Dans une image en niveaux de gris, chaque pixel est noir, blanc, ou a un niveau de gris entre les deux. Cela signifie que les trois couleurs/composantes ont la même valeur.

L'œil est plus sensible à certaines couleurs qu'à d'autres. Le vert (pur), par exemple, paraît plus clair que le bleu (pur). Pour tenir compte de cette sensibilité, dans la transformation d'une image couleur en une image en niveaux de gris, on ne prend généralement pas la moyenne arithmétique des intensités de couleurs fondamentales, mais une moyenne pondérée.

La formule standard donnant le niveau de gris en fonction des trois composantes est :

gris = int(round(0.299·rouge + 0.587·vert + 0.114·bleu))

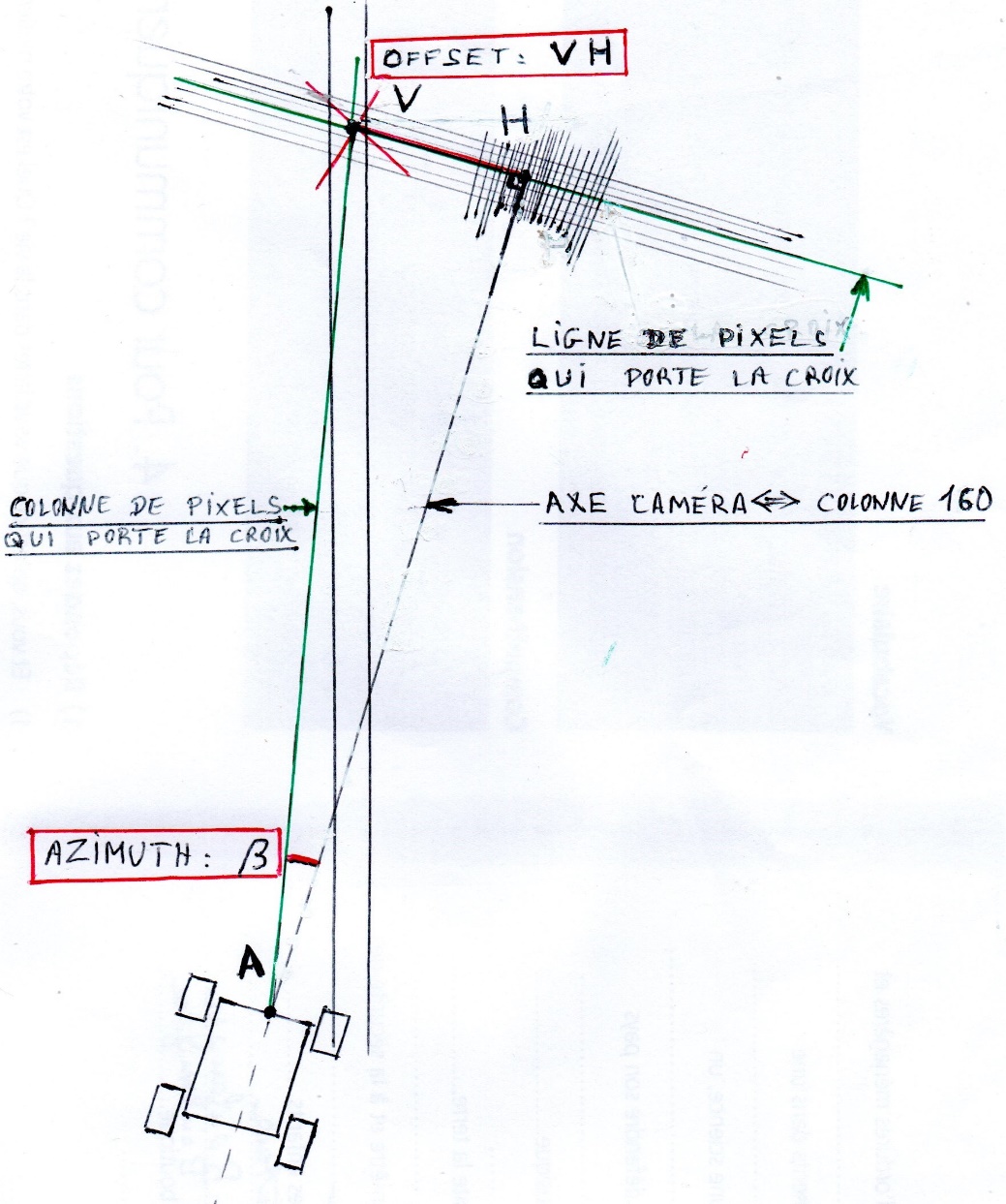
Seuillage :

Le seuillage d'image est la méthode la plus simple de segmentation d'image. À partir d'une image en niveau de gris, le seuillage d'image peut être utilisé pour créer une image comportant uniquement deux valeurs, noir ou blanc (monochrome). On remplace un à un les pixels d'une image par rapport à une valeur seuil fixée (par exemple 123). Ainsi, si un pixel à une valeur supérieure au seuil (par exemple 150), il prendra la valeur 255 (blanc), et si sa valeur est inférieure (par exemple 100), il prendra la valeur 0 (noir).

Avec une image en couleurs, on fera de même avec les trois composantes rouge, vert et bleu. Il y

aura ainsi huit couleurs possibles pour chaque pixel : blanc, noir, rouge, vert, bleu, magenta, jaune et

cyan.

* + 1. **Définitions** :

La figure ci-contre définit :

* Le point V (croix rouge) du « SAUT » noir/blanc
* Le point A pied de la caméra
* Le point H (pied de la perpendiculaire issue de V à l’axe du Truggy et de la caméra)
* **VH est l’offset « visée »**
* **β est l’azimut**

Les images prises par le CCD de la caméra s’inscrivent dans un rectangle de 320 colonnes et 240 lignes pour cette étude. Projetées au sol, on y trouve l’offset l’offset β (fig). La ligne de pixels n° 1 est la plus éloignée du Truggy et la ligne de pixels n° 240 est la plus proche (quasiment au pied de la caméra).

La caméra étant installée au milieu avant du Truggy, l’axe du Truggy et l’axe longitudinal de la caméra sont confondus en pointillé sur cette image.

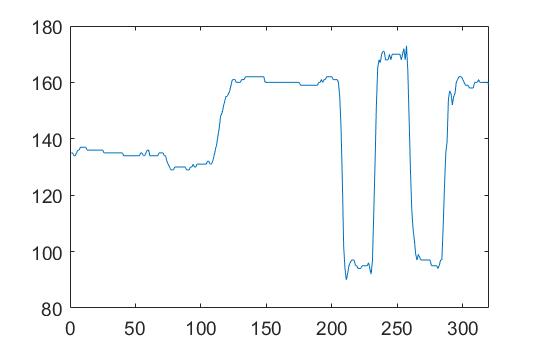
**4.3.2 Principe des calculs de détection du « SAUT » et des offset:**

1. Principe pour détecter le « SAUT » noir/blanc:

On choisit un n° de ligne assez éloignée, par exemple la ligne n° 50 dans la figure ci-dessous.

Chacun des 320 pixels de la ligne 50 a reçu un niveau de lumière (de photons) dans une échelle 0 à 255, dans la figure le niveau de lumière est inférieur à 180 (ordinnée).

Les niveaux d’illumination de chaque pixel de la ligne sont portés sur la courbe ci-dessous.



Le traitement de l’image consiste à détecter le « SAUT » du passage du noir de la bande noire (repérée par les pixels 210 à 230) au blanc de la bande blanche (repérée par les pixels 230 à 260).

* **Voir annexe (à venir) dédiée à ce traitement image qui permet la localisation de ce SAUT.**

Cette localisation est matérialisée sur l’image de la page précédente sous la forme d’une croix blanche.

C’est donc cette croix qui positionne le point V (point « visée »).

Connaissant le numéro du pixel V, ci-dessus correspondant au pixel 230, et celui de H (pixel 160 qui est le milieu des 320 pixels d’une ligne), on en déduit la valeur de l’offset « visée », soit

VH = 230-160 = 70 pixels

Connaissant la largeur du pixel à cette distance on pourra en déduire la valeur de l’offset VH

1. **Calcul de l’offset VH et de l’azimut associé :**

* **Toutes longueurs en cm.**

Nous avons vu au § 3 que les pixels projetés au sol sont déformés. Les pixels réels de la caméra sont des carrés et les pixels projetés au sol sont des trapèzes.

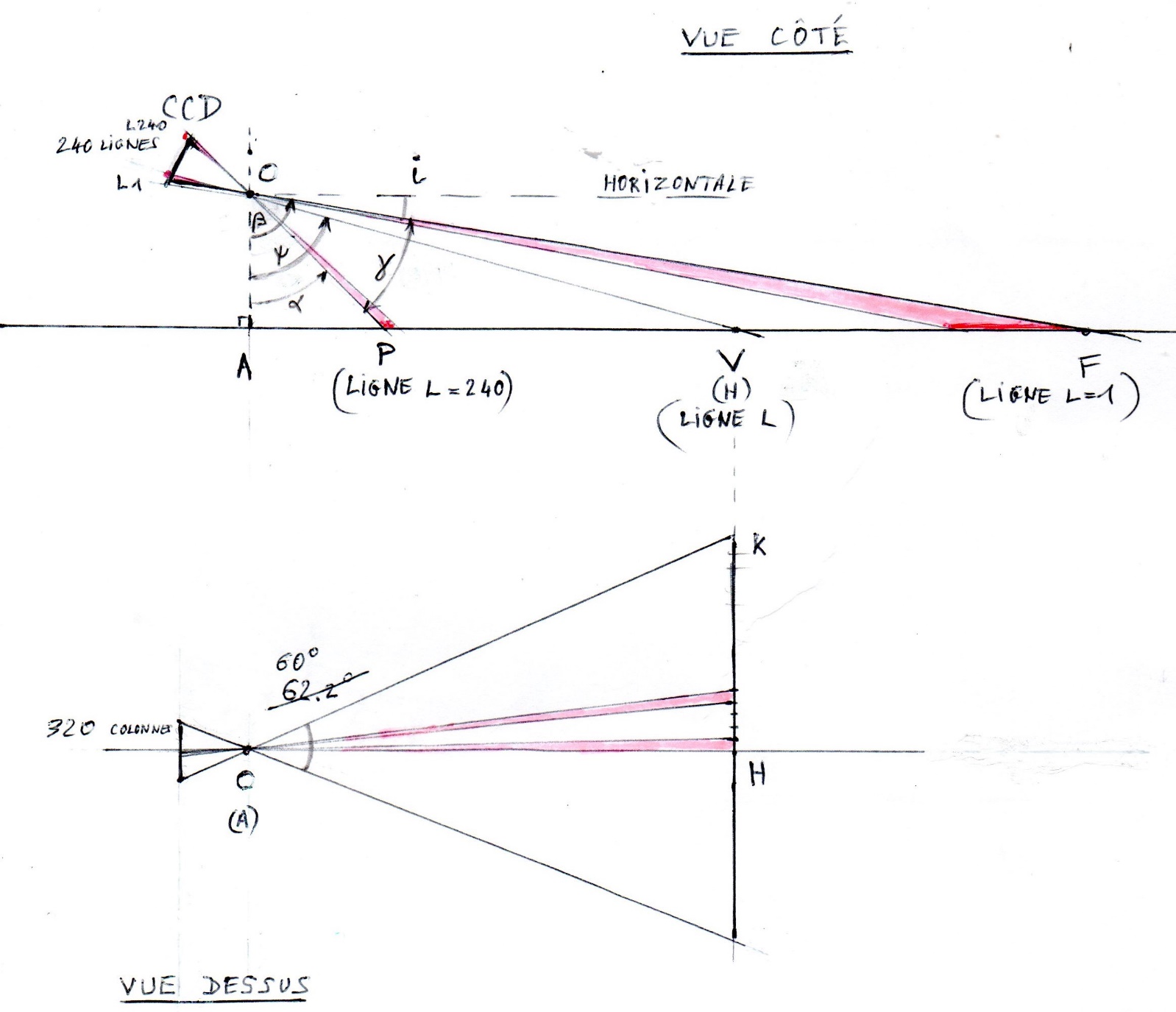
**Pour le format choisi 320 x 240, qui a été binné d’un facteur 10, le pixel objet n’est pas un carré parfait. Mais ne connaissant pas la méthode de binnage, on prend comme hypothèse qu’il a une forme proche du carré.**

Plus la distance de visée est grande, plus les pixels projetés au sol sont grands.

Pour une ligne L donnée, il faut donc d’abord calculer AH, pour calculer ensuite la largeur des pixels de la ligne L considérée.

Les considérations géométriques de position (hauteur) et d’inclinaison de la caméra, et le choix du format permettront donc de calculer les formules des dimensions des pixels.

Considérons les figures suivantes :



**Calage/étalonnage de la caméra** : il est impossible de mesurer l’inclinaison i de la caméra avec la précision nécessaire. Il faut donc étalonner le calage de la caméra.

On inclinera donc la caméra vers le sol d’un angle i (/ horizontale) de manière à ce que la 1ière ligne de l’image vise une barre blanche posée en travers sur le sol à une distance de l’ordre de 5 m de la caméra par exemple, définissant ainsi le point F.

On déterminera ensuite le point P en plaçant une autre barre blanche qui définira la position de la ligne de pixels n° 240. On mesure ensuite la distances AP.

Suivant le même principe on place une barre blanche à gauche de l’axe de la caméra pour qu’elle coïncide avec la colonne n° 1 et une autre barre blanche à droite pour qu’elle coïncide avec la colonne n° 320, ceci pour les lignes 1 et 240.

Puis on mesure les distances entre les barres.

On a donc étalonné le calage de la caméra et on obtient les dimensions de l’image projetée au sol : c’est un trapèze.

Cet étalonnage permet aussi de vérifier les champs de vue donnés par le constructeur de la caméra.

De plus :

* sur la vue de dessus, on voit que les pixels du CCD (plan focal) se projettent sur le plan image au sol avec un agrandissement dépendant du rapport distance caméra-plan image / focale**.**

**Donc au sol (plan image), les « pixels projetés » sont de même largeur**

* sur la vue de côté par contre, on voit que les pixels du CCD (plan focal) se projettent sur le sol en étant agrandis (selon la distance), mais aussi en étant allongés, car l’axe de la caméra n’est plus perpendiculaire au plan image (le sol). Ce sont donc des trapèzes.

**Calcul de la largeur d’un pixel sur la ligne n° L (figure ci-dessus – vue de dessus) :**

Sur la ligne n° L de l’image passant par H on a 320 pixels de même largeur.

On a KH (320/2 pixels) = TAN (62,2/2) \* CH (ou AH en projection)

soit 1 pixel = TAN(31,1)\*AH/160

La largeur larg (cm) d’un pixel de la ligne n° L située à la distance AH est donc :

**larg = 0,00377 \* AH**

**AH en cm (dépend de L) :**

Vérification du champ de vue réel γ en vertical de la caméra (après étalonnage complet):

γ = β – α = ATAN (AF/AC) – ATAN(AP/AC)

* à comparer avec le champ de vue donné par le constructeur)

**Calcul de la distance AH en fonction du n° L de la ligne de pixels (dans le plan du sol) :**

**🡺 ce calcul est fait avec les valeurs de calage caméra « 500 cm » du 1ier juillet :**

AC=27 cm

Il y a 240 lignes entre les points F et P (limites du champ de vue vertical).

**ε** est l’angle qui sous-tend un pixel binné. Pour le format choisi **ε** = 48,8 / 240 = 0,2033 °

AH = AC\*TAN ψ avec ψ = α + ((240-L)\*ε)

AH = AC \* TAN (α + (241 - L) \* ε**)**

**APPLICATION au calage caméra réalisé le 1ier juillet 2020 sur piste Pech David :**

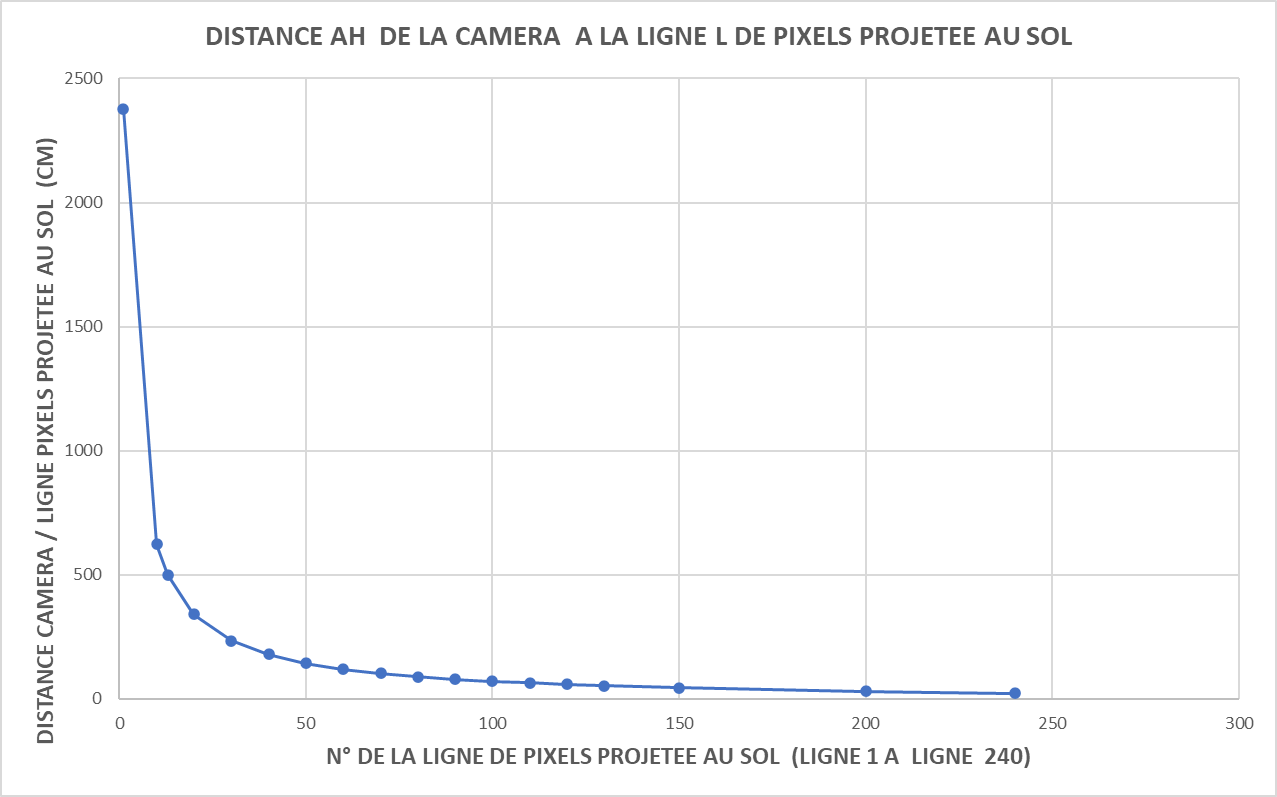
Avec AC = 27 cm et avec bobine blanche localisée à 500 cm du pied de la caméra sur la ligne de pixels 13 :

Angles **en °** et AH en cm

**AH = 27 \* TAN (40,5491034 +(241-L)\* 0,2033333 ) 0,2033333**

Même formule pour calculs en **radians**et AH toujours en cm:

**AH = 27 \* TAN (0,707715363 +(241-L)\* 0,00354884 )**



1. Calcul de l’offset VH :

L’offset visée VH a été calculé en a) en nombre de pixels.

Sa longueur est :

**VH = larg \* nb pixels**

1. Calcul de l’azimuth **β**  :

β est défini en 4.3.1

**β = ATAN (VH/AH) en ° ou rd**

**IMPORTANT :**

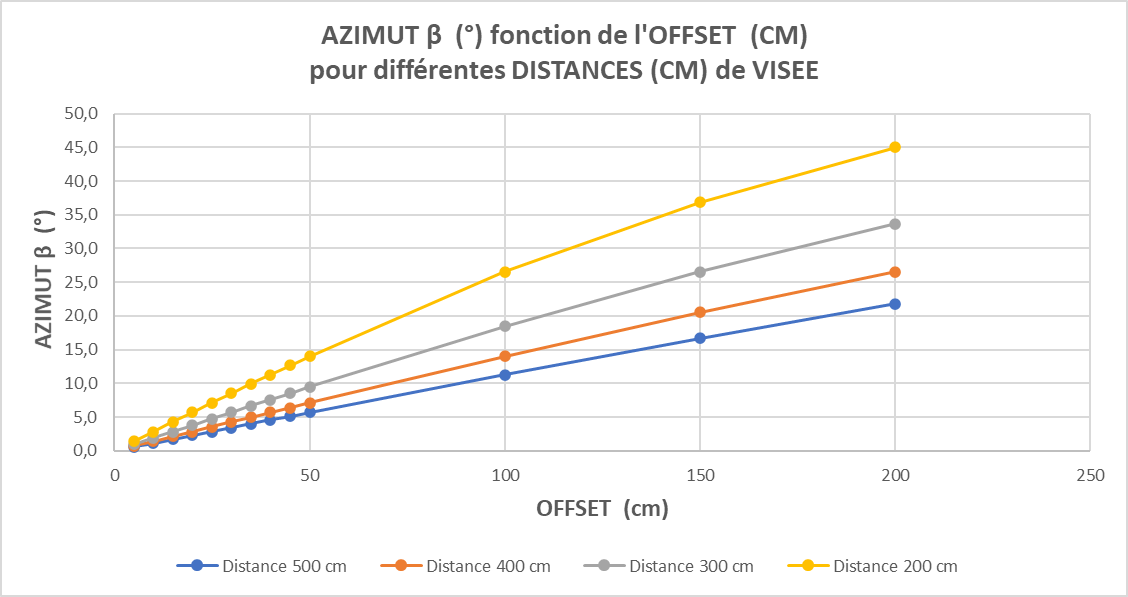
Il faut connaître la position du « SAUT » à droite ou à gauche de l’axe du Truggy et donc faire un test.

Par convention si le(s) pixel(s) « SAUT » est à droite de l’axe du Truggy, on testera :

**. si le n° pixel SAUT > 160, on dira VH > 0 et donc β >0 et V à droite de H et on braquera à droite**

**. si le n° pixel SAUT < 160, on dira VH < 0 et donc β <0 et V à gauche de H et on braquera à gauche**

**Courbes**  **de l’azimut** **β en fonction de l’offset pour différentes distances de visée au sol ;**

****

1. **LOI DE PILOTAGE**

La loi consistant à braquer en permanence (pour chaque valeur reçue du Rasberry, environ toutes les 20 ms) les roues de la valeur de l’angle azimuth **β** est recommandée pour les premiers essais.

1. **CONCLUSION :**

Les croix détectant les « SAUTS » bande noire gauche à bande blanche centrale pourront donc être positionnées par rapport à l’axe du Truggy de manière à commander le braquage β de ses roues.

Le calcul de β doit être fait avec anticipation pour neutraliser au mieux le temps de réponse du servo de la direction du Truggy.

Pour cela, il est nécessaire de connaître et d’optimiser un certain nombre de paramètres :

* Choisir et effectuer le calage de la caméra (hauteur, inclinaison) et faire un **étalonnage précis** du champ (horizontal et vertical) de visée de la caméra
* optimiser le choix du format de l’image : format en largeur pour garder la bande blanche dans l’image, pas trop de binning (pour un contraste meilleur ?), pas trop de pixels pour éviter un temps de calcul trop élevé, …
* l’image initialement en couleurs doit être traitée (OpenCV) avec un passage en noir et blanc et avec le réglage de luminosité/contraste/ … adéquat pour **détecter les « SAUTS » le plus loin possible (anticipation) et avec une fiabilité maximum.**
* …

Mars 2021

**ANNEXE 2 – CALCUL DES OFFSETS VISEE ET SITUATION/PISTE**

Rappels terminologie : voir figures suivantes a à d

* **VH est la distance « offset  visée »**
* **β est l’angle entre l’axe du Truggy et la direction du point V de la visée, nommé « azimut »**
* **γ est l’angle entre les axes piste et Truggy (déviation)**
* **AD est la distance du Truggy à l’axe de la piste (ou « écart Truggy »)**

Dans l’annexe 1 (§ 5) on avait recommandé pour les premiers essais une loi de pilotage qui consistait à commander toutes les 20ms environ un braquage β1 des roues égal à l’azimut calculé (fig a).

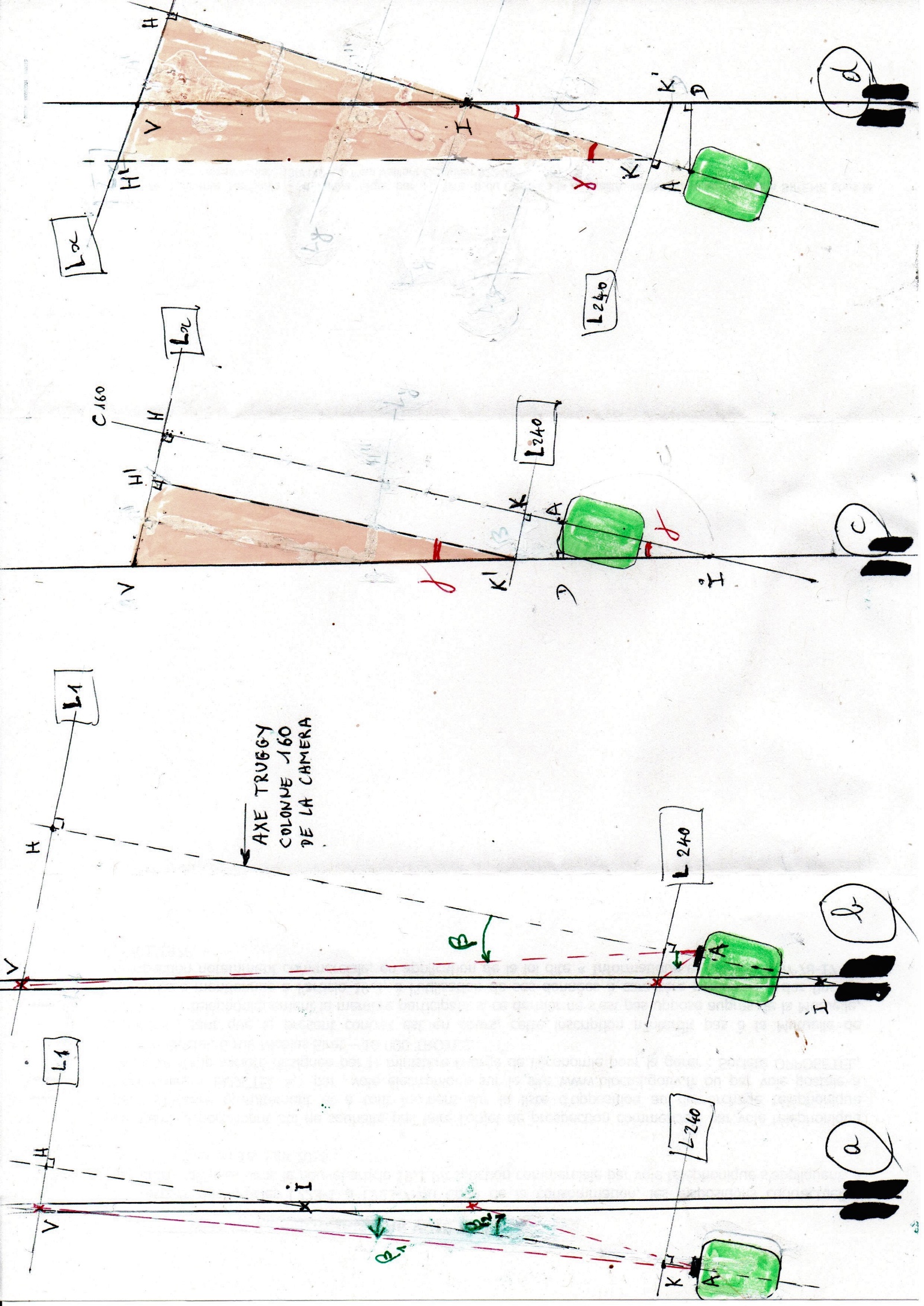
Un problème apparaît avec cette méthode :

La détection de la croix se fait actuellement de manière séquentielle sur les lignes 25, 35 et 45, puis s’il n’y a pas détection, sur les lignes 100, 14 et 180.

Quand le véhicule se rapproche de la piste (fig a) il arrive que le point visé V est parfois situé en dessous du point d’intersection I, **auquel cas l’angle de braquage devient β2 de sens opposé à β1** (cas où la détection de la croix se fait sur une ligne de rang supérieur). Dans ce cas **on braque trop** par rapport au cas où le point visé V est plus loin.

Quand le véhicule s’éloigne de la piste (fig b), le point visé V est toujours au dessus du point d’intersection I, l’angle de braquage est toujours de même signe, d’autant plus important qu’on détecte trop près.

**Comme on ne sait pas à quelle ligne on aura la croix, et compte tenu de la fréquence de rafraîchissement** (de 25 à quelques Hz ?) et du temps de réponse de la commande de braquage, les premiers essais menés sur piste en ligne droite ont montré des louvoiements et des divergences amenant le Truggy à quitter la piste.



1. **Optimisation de la loi de pilotage actuelle :**

Vu qu’actuellement les angles de braquage paraissent trop importants, induisant des louvoiements importants et une divergence du pilotage, on pourra tester une loi où l’angle de braquage sera atténué, c’est-à-dire une fraction de l’angle β.

On pourra tester β/2, β/3, autres …

1. **Nouvelle loi de pilotage avec γ et AD**

L’expérience faite avec un gyromètre délivrant avec une fréquence élevée (> 80 Hz ?)

l’angle γ et une loi de pilotage simple (proche de PID) avait donné un très bon comportement en ligne droite.

* **Une loi de pilotage utilisant comme entrées les données γ (et AD ?) devrait donc permettre d’améliorer les choses.**

Les figures c et d montrent que l’on peut, par traitement des images, extraire rapidement les valeurs de γ et AD.

Une loi de pilotage telle que celle testée sur le Truggy avec le gyro et avec l’utilisation de la donnée offset Truggy (AD) devrait permettre d’obtenir un bon pilotage, en ligne droite et en virage.

**Cas de la figure c :**

Calcul de γ :

Triangle VH’K’: **γ = ATAN (VH’/H’K’)**

avec VH’= VH-KK’ (VH différence des pixels sur ligne Lx et KK’ diff pixels sur ligne L240)

et K’H’=KH (différence des pixels sur colonne 160)

Calcul de AD (fig c) :

On pose IA = x ;

On a TAN γ = VH/IH d’où on tire IH

D’où x = IH – AH avec AH = AK (connu ≈ 17 cm) + KH (différence pixels Lx-L240)

D’où **AD = x . sin γ**

**Cas de la figure d :**

Calcul de γ  :

Triangle KHH’ : **γ = ATAN (HH’/HK)**

avec HH’ = somme des pixels VH ligne Lx et des pixels KK’ ligne 240

et HK = différence des pixels colonne 160

Calcul de AD :

On pose IA = x ;

On a TAN γ = VH/IH d’où on tire IH

D’où x = -(IH -AH) avec AH = AK (connu ≈ 17 cm) + KH (différence pixels Lx-L240)

D’où **AD = x . sin γ**

🡺 **On voit donc que :**

* **on peut donc calculer la déviation γ et piloter le Truggy comme avec un gyromètre (déjà testé) avec la donnée γ et la distance AD (offset Truggy) qui permet de situer le véhicule par rapport à la piste. Cette donnée supplémentaire est sans doute très utile pour consolider la loi de pilotage**

**🡺 Attention : les formules de calcul de γ et de AD sont identiques pour les 2 cas de figures (c et d), à condition de donner un signe + ou - aux distances droite/gauche mesurées sur les lignes L, l’origine étant la colonne 120.**

**🡺 Point très important :**

* **pour calculer γ, cette loi de pilotage ne nécessite pas d’aller chercher des croix sur les premières lignes les plus éloignées. Donc elle est probablement plus rapide et plus robuste.**

**ANNEXE 3 – CARTOGRAPHIE DE L’IMAGE AU SOL** Avril 2021

1. **Portée maximale de la caméra :**

Plusieurs essais ont été menés pour rechercher les paramètres optimaux pour l’obtention d’une croix de détection de la piste blanche la plus éloignée possible : **c’est la portée réelle utile de la caméra.**

Voir une des images obtenues annexe 4.

Conditions d’essais :

. La nouvelle piste (transportable) est neuve, le ruban adhésif blanc ou noir est brillant donc spéculaire, mais c’est le même que celui de la TRR donc il est parfaitement représentatif.

. Journée ensoleillée, sous un préau, donc à l’ombre pendant les essais. La luminosité ambiante est toutefois assez forte, ce qui induit malgré tout des réflexions parasites sur les bandes blanche et noire de la piste.

. Plusieurs mesures ont été faites en début et en fin d’après-midi (luminosité moins forte).

Les paramètres suivants ont été bougés :

* Brightness (luminosité)
* Contraste
* Hauteur du saut
* Nb de pixels
* Sens du Truggy sur la piste
* …

Résultats :

* **La portée rélle utile de la caméra, c’est-à-dire quand la ligne 25 positionne la croix sur la piste blanche, est de un peu plus de 2 m dans** les conditions d’environnement plutôt difficiles (sous un porche, luminosité latérale, sol légèrement brillant, …).
* Positionner un verre polarisant devant la caméra permet d’éviter d’éventuelles réflexions parasites, et donc dans certaines circonstances de détecter la croix sur la ligne 25 au lieu de 35 ou 45. Mais il faut augmenter la brightness (le verre polarisant des lunettes de soleil fait perdre de la luminosité).
* **La détection de la croix est d’autant plus facile/fiable qu’on vise de moins en moins loin**

1. **Déformation de l’image projetée au sol :**

On conçoit que l’image projetée au sol soit déformée car **elle est projetée sur une surface quasiment perpendiculaire au plan du CCD**, de plus l’objectif est une petite lentille simple de qualité médiocre.

🡺 Il faut donc déterminer le contour et les dimensions de l’image projetée au sol pour pouvoir ensuite traiter géométriquement de manière correcte cette image.

Jusqu’à présent on n’avait pas pris en compte les distorsions induites par l’objectif de la caméra.

On considérait que l’image projetée au sol était parfaite, c’est-à-dire de forme trapézoïdale (figure ci-après, croquis en rouge). En réalité, en plaçant des repères au sol le long du contour de l’image, sur la figure le point I par exemple, et en mesurant leur position x,y, on voit que le contour de l’image au sol est déformé (figure ci-après).

***Le point I mesuré au sol apparaît en I’ sur l’image de la caméra alors qu’il est en réalité plus près de la caméra.* *Les lignes de pixels les plus éloignées, soit les lignes 0,1,2,3,4,5,6, … sont des courbes qui se déforment et « rapprochent » les objets, d’autant plus que l’on s’éloigne de l’axe optique (ou axe Truggy).***



En conséquence, le point V sur l’image est en réalité situé en V’, le nombre de pixels entre V et H est le même que celui entre V’ et H’, mais leur largeur réelle sur V’H’ est légèrement réduite (de cos γ en première approche).

Conséquences sur les calculs des angles β (option 1 de pilotage) et γ (option 2 de pilotage) :

Ces angles ont été calculés jusqu’à présent à partir de longueurs de pixels dans le cas d’une projection au sol sans déformation. En réalité donc, les lignes et les colonnes de pixels au sol sont déformées (mais le nombre de pixels est invariant).

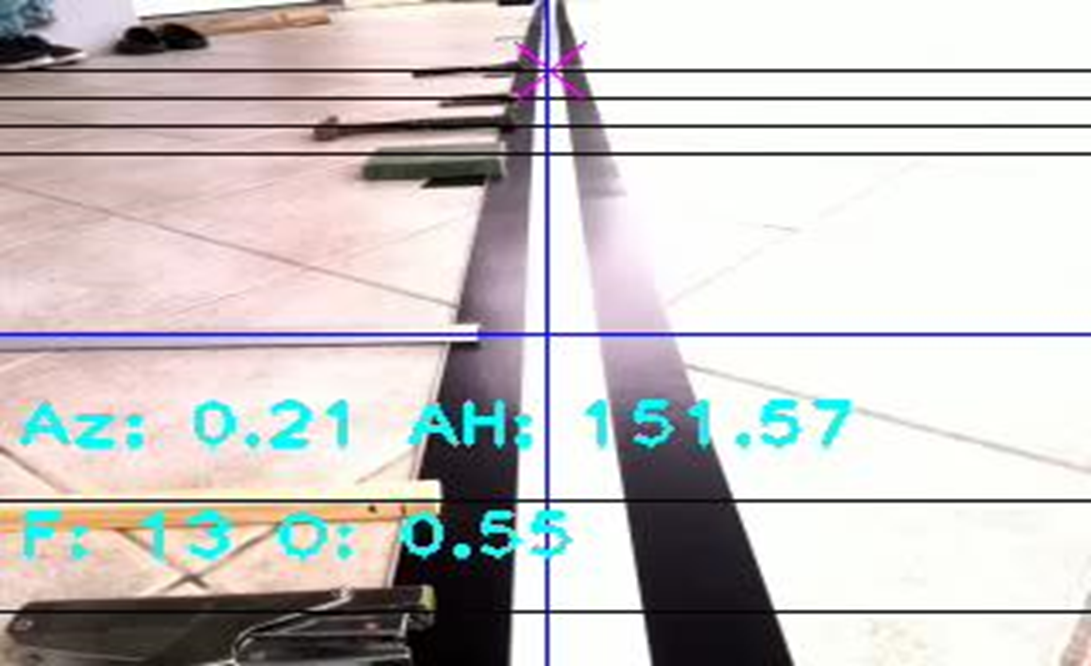
Les pixels sont simplement légèrement plus étroits par rapport à l’image trapèze (en rouge) non déformée, au fur et à mesure que l’on s’éloigne de l’axe de la caméra. Mais la correction en cos à faire est négligeable : pour des angles β ou γ petits, par exemple γ=20°, cos 20° = 0.94 , l’erreur reste négligeable pour nos applications.

De même pour la distance V’H’ plus courte du facteur cos que VH.

Il paraît donc peu nécessaire de reprendre tous les calculs de ces angles, ce qui alourdirait le traitement de l’image, alors que le but du pilotage est de garder des angles β ou γ petits. (A Confirmer)

**ANNEXE 4 – CALAGE DE LA CAMERA ET FORMULES APPLICABLES** Avril 2021

1. **Etalonnage de l’image au sol de la caméra : image prise le 28 avril 2021**

****

*Rappel : le CCD est utilisé avec le format 320 colonnes x 240 lignes (format binné).*

En partant du haut de l’image :

* la bordure/ligne supérieure de l’image est la ligne de pixels n° 0
* les lignes horizontales noires sont les lignes 25,35,45,55,65,75
* la ligne bleue horizontale est la ligne 119 ou 120
* les 2 lignes noires du bas sont les lignes 180 et 220
* la bordure inférieure de l’image est la ligne de pixels n° 239

1. **FORMULES (pour le calage de la caméra à 5,5 m conforme à l’image ci-dessus du 28 avril 2021 :**



Données géométriques mesurées :

* calage caméra : **ligne 0 à 5,5 m de la caméra (point F),** ligne 239 à 18 cm de la caméra (point P)
* largeur de l’image sur la ligne 239 : 36 cm, hauteur de la caméra par rapport au sol : 28 cm
* champ horizontal mesuré : **60 °** , champ vertical mesuré : **54,3 °**
* piste : bandes noires et blanche de 5 cm de largeur, soit 15 cm de largeur totale

**🡺 ces nouvelles données sont prises en compte pour mettre à jour toutes les formules de l’annexe 2, nécessaires au traitement de l’image.**

**Calcul de la largeur d’un pixel sur la ligne n° L :** (voir fig § 4.3.2.b de annexe 1)

AH (ou AV) est la distance entre A et la ligne L

1 pixel = TAN(60/2)\*AH/160

**larg 1 pix = 0,0036084392 \* AH**

(cm)

**Calcul de la distance AH en fonction du n° L de la ligne de pixels (dans le plan du sol) :**

Il y a 240 lignes entre les points F et P (limites du champ de vue vertical).

Champ vertical caméra**:** 54,353526648 °et **ε** est l’angle qui sous-tend un pixel binné.

Donc **ε** = 54,35352664 ° / 240 = 0,2264730277 ° soit

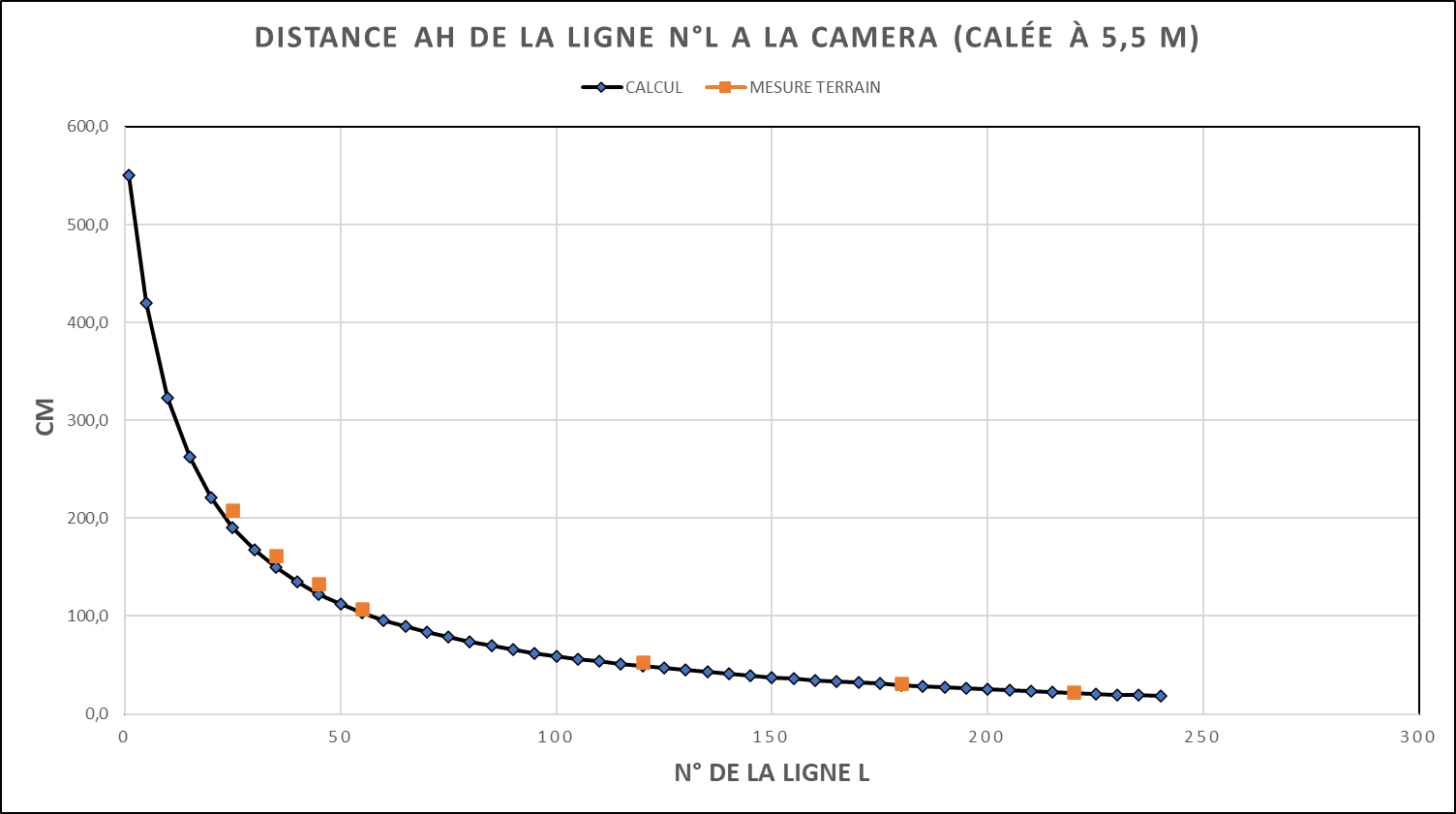
AH = AC\*TAN ψ avec ψ = α + ((240-L)\*ε) et α = 32,73522 °

Soit AH = AC \* TAN (α + (241 - L) \* ε**)**

**AH = 28 \* TAN (32,73522 +(241-L)\* 0,2264730277 ) 0,2033333**

(angles en °) (cm)

Voir la courbe correspondante ci-après :

****

**Les points en rouge** sur la courbe correspondent aux distances (mesurées avec un mètre) des objets placés (une personne devant la vidéo guide une autre personne qui place les objets) sur les lignes 1, 25, 35, 45, 55, 120, 180 et 220.

NOTA : très bonne concordance des valeurs des distances. L’optique ne déforme pas les pixels dans l’axe de la caméra.

**Calcul de l’offset visée VH (figure p 18) :**

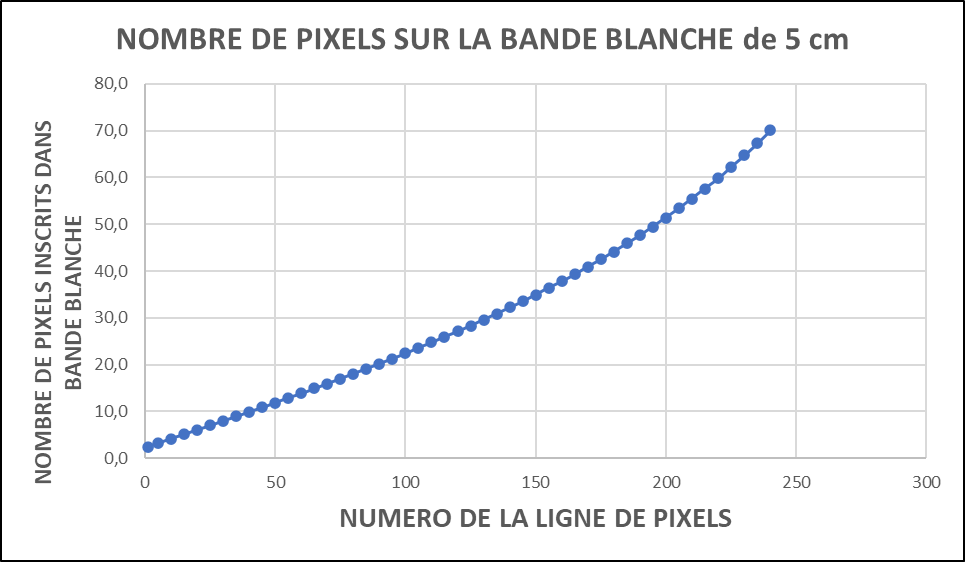
(cm)

**VH = larg \* nb pixels**

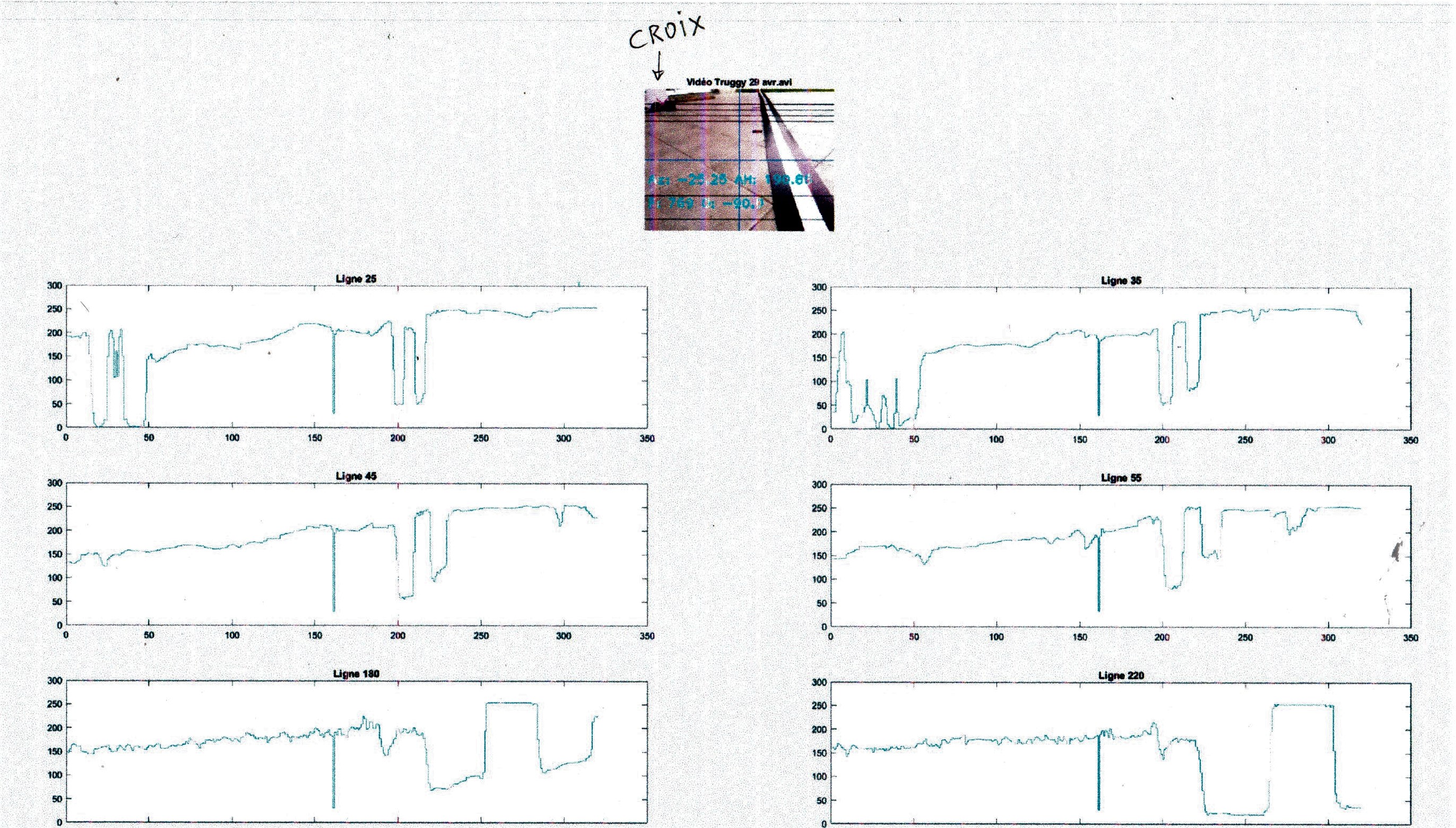
**Calcul de l’azimuth β (selon figure annexe 2) :**

**β = ATAN (VH/AH) en ° ou rd**

**Largeur et Nombre de pixels sur la bande blanche de 5 cm en fonction du n° de la ligne L :**

****

La figure suivante montre les spectres des 6 lignes d’une image de la vidéo du 29 avril 2021. Ces spectres représentent l’intensité lumineuse de chacun des 320 pixels de la ligne considérée (25, 35, 45, 55, 180, 220). On remarque les signatures différentes des bandes blanche et noires selon les lignes.



**CONCLUSION COMMUNE AUX 4 ANNEXES :**

**La caméra Truggy a été calée le 28 avril 2021. Suite à ce calage, les formules de calcul des offsets et des angles ont été mises à jour dans l’annexe 4.**

**Options de pilotage envisageables :**

* **Option 1: on calcule l’azimut β, puis on braque les roues pour annuler β, ce qui dirige le Truggy vers le centre de la piste visé à 3 à 4 m (ordre de grandeur souhaitable/réaliste) en avant du Truggy**
* **Option 2 : on calcule l’azimut γ (vecteur Truggy) et « l’écart Truggy »**
* **Option 3 ( ?) : on utilise les « offset visée » et « écart Truggy » pour les annuler et maintenir ainsi le véhicule sur la piste blanche**

**ANNEXE 5 – OPTIONS DE PILOTAGE – CHOIX**  juillet 2021

1. **EXPLOITATION DERNIERS ESSAIS DU TRUGGY :**

Les derniers essais du TRUGGY sur piste ouverte et surtout sur piste couverte ont montré qu’un pourcentage plus ou moins important de fausses croix perturbent fortement ou interrompent les essais.

**Sur une ligne droite, la parade a été trouvée en empêchant tout braquage intempestif (braquage borné).**

Pour les virages, notamment les virages « serrés » du Roselab (braquage β = ± 15,6 ° pour le premier virage), on ne peut pas borner le braquage, ou on le borne avec des valeurs de l’ordre de 17 °.

Donc :

* si la croix est vraie, et relativement loin on braque fortement (courbure piste forte). Si on « vise » plus près, on sait qu’on n’anticipe pas assez, donc on ne peut que rouler très lentement en louvoyant
* si la croix est fausse, c’est pire

Au moins 2 constatations s’imposent :

1. **il faut de toute façon réduire ou éliminer l’occurrence des fausses croix**
2. **il faut un pilotage adapté aux courbes.**

Concernant le pilotage en courbes, il faut réfléchir à la solution déjà testée au Fablab :

1. à l’entrée des 2 virages en bout de piste (figure) , on impose un braquage prédéterminé. Le véhicule accomplira le ½ cercle du virage, avec une erreur « transversale » (ou offset Truggy) à la sortie, qu’il faut minimiser.

Pour minimiser cette erreur, on peut réfléchir à :

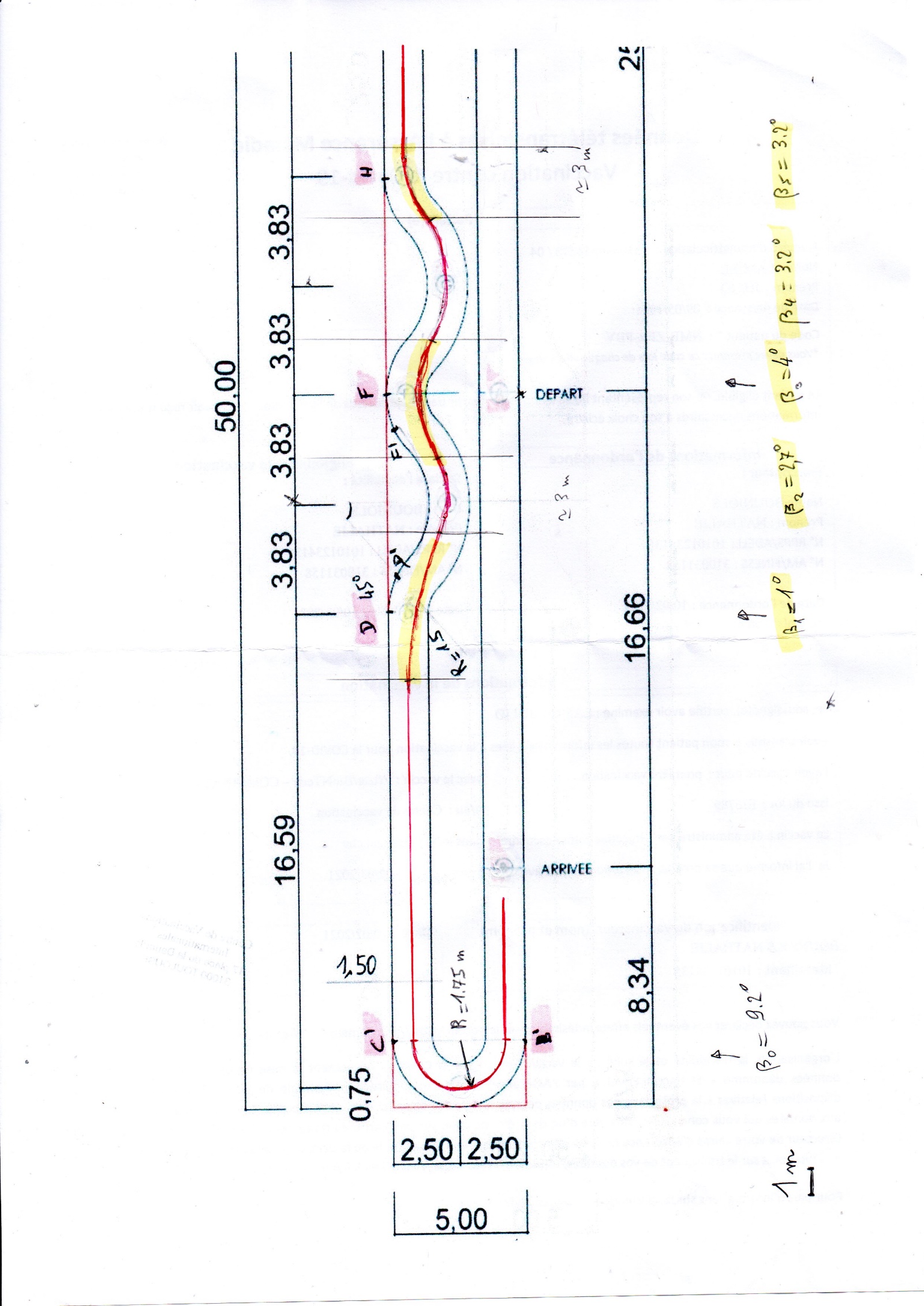
. exploiter une croix sur une ligne proche (par exemple 230 – donc à priori croix fiable) pour récupérer l’offset de position du Truggy par rapport à la piste blanche

. corriger le braquage en temps réel, tout en restant à l’intérieur de 2 bornes serrées : par exemple le braquage pour ces 2 virages est imposé à β = + 9,2 °, et l’offset mesuré peut corriger en temps réel ce braquage entre β = 8,5 et 10 °.

Concernant les chicanes, même procédure :

1. la partie chicane est découpée en 5 tronçons (figure). On pourra cependant tenter « d’adoucir » les braquages en ne suivant pas strictement la piste blanche.

Dans ce cas on aura des braquages successifs de β = ± 1 à 4° seulement ce qui permet de passer plus vite ?



1. **OPTIONS DE PILOTAGE:**

Piloter le Truggy, c’est le diriger, lui communiquer des mouvements.

Pour la TRR, catégorie « roulants », il faut partir de la ligne de départ et rejoindre la ligne d’arrivée en se déplaçant entre 2 bordures. En plus des bordures, une bande blanche encadrée par deux bandes noires peut être utilisée pour se guider.

**🡺 le guidage par la bande blanche est la solution retenue pour guider le Truggy à l’aide d’une caméra.**

*Comment naviguer (connaître la position du Truggy, rejoindre un point de coordonnées connues, calculer les distance et vitesse de déplacement, ..) et piloter le Truggy ?*

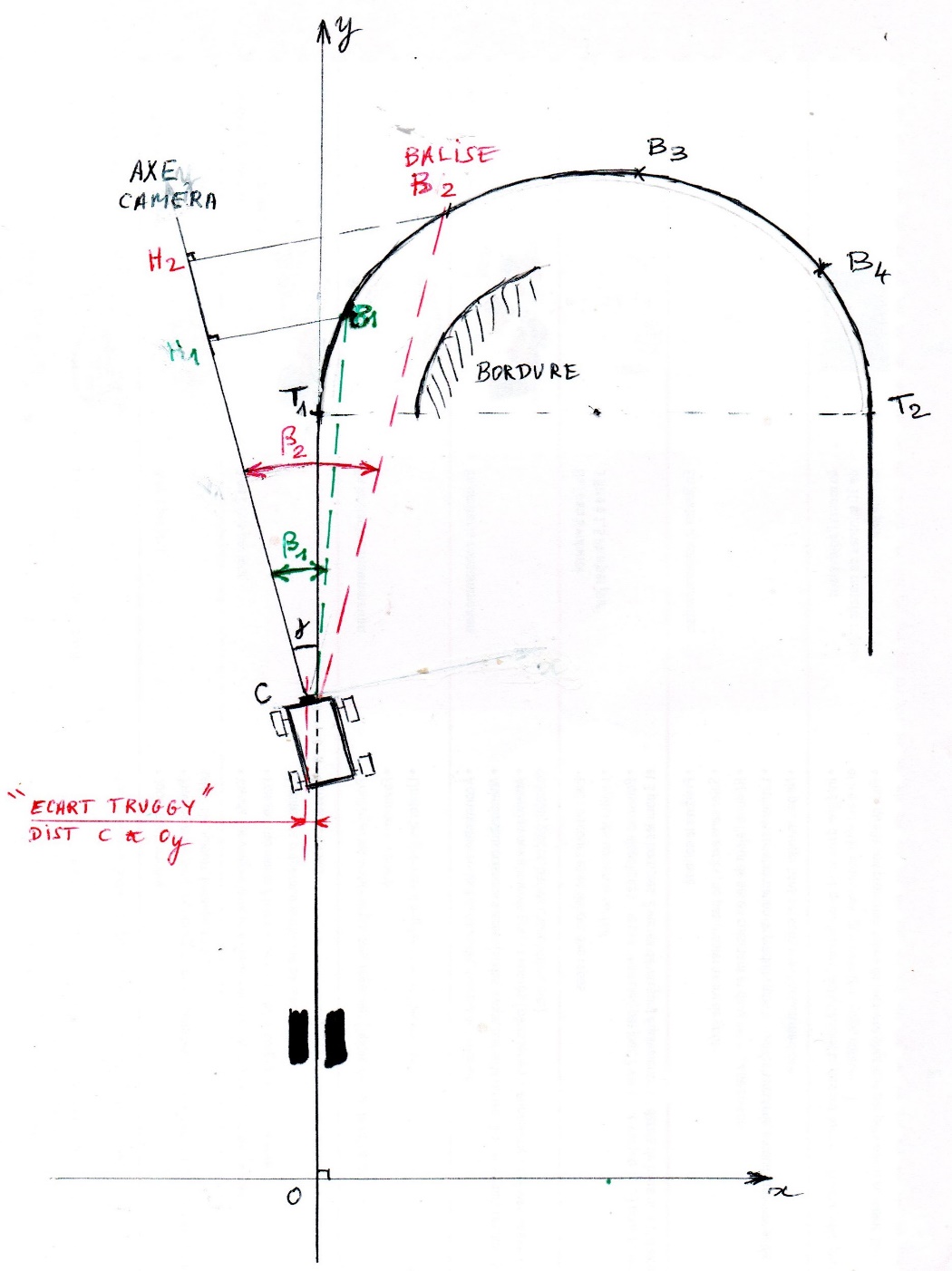
**La piste de la TRR :**

Les dimensions de cette piste sont connues (voir règlement).

Le milieu de la piste blanche est une courbe constituée de segments de droites et de cercles.

La figure suivante montre une portion de cette courbe (depuis le point de départ O jusqu’à la sortie du 1Ier virage).

**🡺 cette courbe, appelée « trajectoire idéale » sera modélisée dans le repère orthonormé Oxy.** On pourra donc déterminer facilement, si besoin, les coordonnées x,y de chaque point, par exemple V1, de la courbe.



**2.1 PILOTAGE EN LIGNES DROITES :**

Option D1 : Pilotage avec braquage suivant l’azimut β :

Cette option a donné des résultats satisfaisants. L’effet des fausses croix, donc des faux braquages, est annihilé par un bornage du braquage β = 0° +/- 2° environ.

Option D2 : Pilotage avec azimut γ et/ou écart Truggy :

Option non encore testée. L’azimut γ et l’écart Truggy seront calculés en exploitant les lignes les plus basses des images, ce qui diminuera potentiellement les fausses croix.

Le pilotage avec le seul « écart Truggy » est peut-être suffisant : **à tester**

**2.2 PILOTAGE LORS DES TRANSITIONS DROITE/VIRAGE ET EN VIRAGES:**

Option V1 (V comme Virage): Pilotage avec braquage théorique imposé et correction par « écart Truggy » :

On a testé (piste Fablab Artilect) les virages à braquage constant : satisfaisant.

Le Truggy ralentit fortement à la fin de la ligne droite pour aborder le début T1 du virage.

La transition droite/virage se fait en passant de la loi de pilotage en ligne droite à une loi virage, nommée V1, imposant un braquage fixe et constant, qui va permettre de suivre la trajectoire (1/2 cercle) théorique du virage.

Cependant la trajectoire circulaire obtenue n’est pas toujours parfaitement superposée à la trajectoire idéale du ½ cercle, un « écart Truggy » pouvant se créer à l’entrée du virage (point T1) et donc un « écart Truggy » est possible à la sortie (point T2) (figure précédente).

Il faudra donc faire des corrections le long du virage en utilisant « l’écart Truggy » mesuré en temps réel par la caméra pour corriger la trajectoire du Truggy et ainsi essayer de rester sur la trajectoire idéale et aborder la ligne droite qui suit avec un « écart Truggy » le plus petit possible (20 cm max ?)

🡺 **à tester, et voir la vitesse maximale accessible**

Option B1 (B comme Balise) : Pilotage avec balises :

L’idée est d’utiliser la connaissance de la géométrie de la piste, donc la connaissance précise et parfaite des coordonnées (dans le repère Oxy) de tous les points de la trajectoire idéale.

balises) pour viser un point de type balise (par exemple point B1 figure précédente).

Connaissant les coordonnées de C (grâce à la caméra et à l’odométrie) et de la balise B1, on calcule rapidement la pente de la droite CB1 (avec une grande précision), qu’on additionne à γ (calculé par ailleurs) d’où l’angle azimut β1, et le braquage β1 équivalent.

Cette option T2 est plutôt fiable, soit la fiabilité du γ calculé, correcte quand on cherche des croix sur les lignes basses de l’image.

**Elle présente l’avantage de « viser » un point fixe par définition juste (sans fausse croix).**

La trajectoire suivie par le Truggy est une succession de petits segments de cercles T1B1, B1B2, B2B3, …. Le nombre de balises est à optimiser par tests.

Cette option paraît intéressante aussi pour traverser les chicanes ?

🡺 **à tester, et voir la vitesse maximale accessible**

**CONCLUSION DE L’ANNEXE 5**

La connaissance des coordonnées de « points balises » de la piste TRR n’avait pas été exploitée dans les annexes précédentes.

L’utilisation de « points balises », dont le nombre et la position restent à choisir peut améliorer les performances des lois de pilotage.

🡺 pour les lignes droites, la loi de pilotage D1 semble opérationnelle et préférable, à quelques réglages près, notamment ne pas viser trop loin pour ne pas multiplier les fausses croix

🡺 pour les virages et chicanes (portions de cercles), les lois de pilotage V1 et B1 sont à **tester**.

.

**Remarque :**

La connaissance des coordonnées des points de la piste est une donnée qui permettra de viser des « point balises » judicieusement positionnés.

**L’utilisation systématique de la loi B1 tout au long de la piste pourrait même être envisagée**. Dans ce cas, on piloterait le Truggy en visant plusieurs « points balises » d’une trajectoire (piste blanche) prédéfinie.

Cette méthode se rapproche des solutions d’apprentissage de l’IA (reconnaître une trajectoire «apprise » et naviguer pour la rejoindre).

(*Avec notre IN (intelligence naturelle), on devrait faire la même chose non ?)*