# DRONE – E.L.V.I.S

M1B Guez – Vieloszynski –Baudoin

# Remerciements

Au terme de ce travail, nous exprimons nos vifs remerciements aux personnes ayant contribué, de près ou de loin, à la réalisation de ce projet. Nous souhaitons tout d'abord remercier notre encadrant Mme Roullot qui nous a guidé tout au long de la réalisation de notre travail.

Enfin nous tenons à remercier l'ensemble du corps enseignant de la filière informatique dont Mr Herry et Mr Robin pour leur aide et conseils sur toute la partie programmation.

# Sommaire

# I. Contexte

## A. Introduction

Le but de notre projet est de réaliser un déplacement autonome du drone par identification de flèches au sol. Nous avons utilisés pour cela différentes méthodes de détection qui travaillaient avec des images de références préenregistrées et des images prises par le drone.

Notre projet prend en compte l'étude du traitement d'images en C/C++ et le contrôle du drone avec NodeJs (JavaScript).

Nous avons fait dans un premier temps l'étude théorique tout en créant notre programme en C++ avec OpenCv. Pour trouver les différents seuils de corrélations que nous avions besoin pour la reconnaissance d'images se qui fut notre premier piste ainsi que le redimensionnement suivant la distance de vol du drone par anticipation, nous avons fait différentes acquisitions de valeurs de seuils avec des formes différentes (rectangle, triangle, flèche) pour tester la robustesse de l'algorithme.

Et pour finir nos recherche nous avons cherché d'autre méthodes de détection qui on peu nous aiguiller ver une optimisation de notre détection d'image.

La deuxième partie de notre projet consista à appréhender le drone, en apprenant à le contrôler en NodeJs (JavaScript) puis en faisant la correspondance NodeJs / C-C++.

## B. Cahier des charges

- Traitement d'images :
  - Etude des niveaux de seuils en utilisant la méthode de corrélation (première méthode étudiée).
  - Etude de méthode d'isolement formes géométriques et détection angles
  - Etude de la détection de couleur
  - Etude de la détection par point d'intérêt
  - Faire un programme avec OpenCv permettant d'appliquer ces différentes méthodes.

- Le drone : Node Javascript : Piloter et métriser le drone avec son langage de programmation.

- Réaliser une Interface de récupération de donnée s (images) entre le javascript et notre programme en C/C++

- Faire correspondre toutes ces étapes ensemble pour que le drone devienne autonome.

# II. Etat de l'art

## A. Introduction

Nous avons étudié plusieurs méthodes de détection qui nous ont permis de voir les différentes méthodes possible afin de réaliser notre projet. Nous avons commencé sur une première solution qui utilisait la corrélation pour le traitement d'images, puis d'autres méthodes.

## B. Les différentes méthodes de détection

Ceci énumère les différentes méthodes rencontrées qui seront prochainement plus détaillé :

- Méthode de détection par corrélation :

Cette méthode consiste à corréler deux images : une image source et une image de référence. L'image de référence (image modèle) est préenregistrée dans notre programme en C, le drone fait l'acquisition d'images par flux vidéo et les compares afin de garder une seule image qui lui indiquera le déplacement à effectuer.

Cette méthode prend du temps car le programme effectue 4 corrélations.

- Méthode de détection par formes géométriques et angles :

Cette méthode consiste à détecter le centre d'une forme (cercle) à l'intérieur de la flèche afin de pouvoir tracer une droite passant par le milieu du cercle et de la flèche (sommet). Cette droite sera notre référence, et une seconde droite sera tracée suivant l'angle qu'a la flèche avec celle-ci.

Cette méthode est beaucoup plus rapide car le temps de calcule et le nombre de pixel analyser est réduit.

- Méthode de détection par couleurs :

Cette méthode consiste à isoler certaines couleurs ciblées par l'utilisateur de telle sorte à détecter les couleurs qui présente un intérêt pour cibler des objets particuliers dans une image.

Méthode beaucoup plus rapide, mais qui peu comporter des inconvénient selon la lumière au quelle l'image est soumis.

- Méthode de détection par point d'intérêt :

Cette méthode consiste à détecter des points locaux (régions) qui semblent pertinents sur une image de référence (généralement des angles ( borde ) et ainsi de faire la reconnaissance point à point pour la reconnaitre si l'image de référence se trouve dans l'image saisie .

Méthode assez rapide car elle ne compare que quelques points d'une image.

## C. Schéma

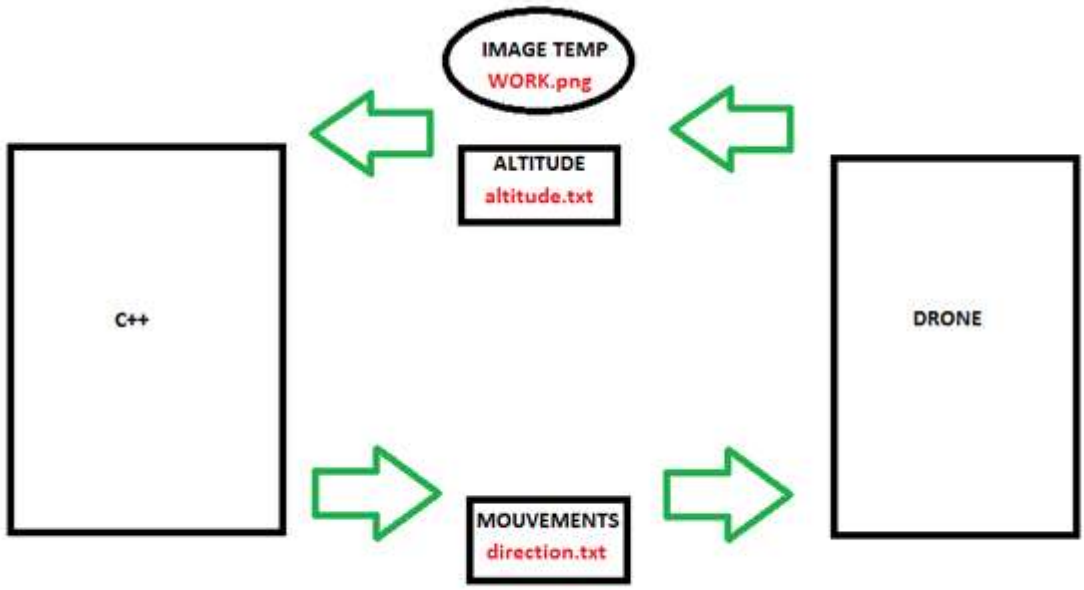



**Figure 1**

Voici un schéma (figure 1) simplifié pour comprendre comment nous allons résoudre notre projet. D'un coté notre programme en c++ utilisant openCv va communiquer avec la partie Drone en Node JavaScript.

Chaque méthode étant différemment énumérée dans la partie précédente, plusieurs paramètres retournés sont en commun à chacune des méthodes :

- L'envoi d'une action de mouvement de notre programme vers le drone
- La récupération de l'altitude grâce à l'altimètre dans le drone vers notre programme
- La récupération d'une image Temp avec le flux vidéo de la caméra ventrale du drone.

## D. Le drone

figure.1

L'AR.Drone est un hélicoptère quadrirotor qui peut se piloter avec un appareil sous iOS, Android ou Symbian (téléphones Nokia) via une liaison Wi-Fi. Il est principalement dédié au divertissement mais dispose d'équipements sophistiqués tels qu'une caméra frontale pour le pilotage, une seconde verticale pour la stabilisation, un accéléromètre trois axes, deux gyroscopes, deux émetteurs récepteurs à ultrasons permettant de calculer l'altitude, de nombreux capteurs ainsi qu'un ordinateur embarqué fonctionnant sur noyau Linux.

L'autonomie est de 12 minutes environ pour un temps de charge d'1h30, il a une vitesse de pointe max de 5m/s et acquiert approximativement 25 images par secondes. (Voir représentation figure 1).

Concernant le développement, l'entreprise Parrot a mis à disposition d'un SDK afin de permettre aux utilisateurs confirmés de développer de nouvelles fonctionnalités pour l'AR.Drone.

# III. Traitement d'images

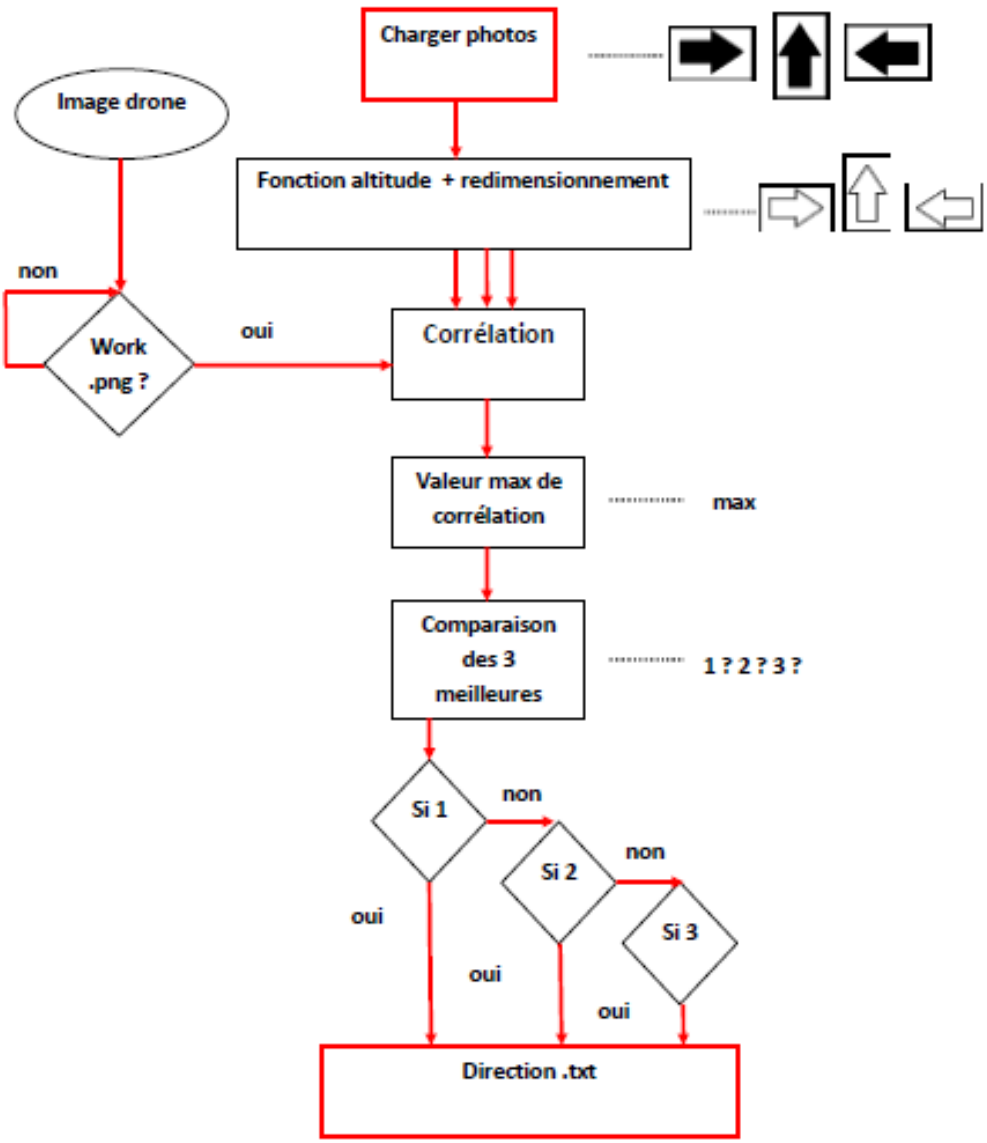
## A. Introduction

Nous avons commencé par étudier les seuils de détection pour connaitre la distance ou la flèche était la mieux reconnue.

## B. Détection par corrélation

Cette méthode consiste à corréler deux images : une image source et une image de référence. L'image de référence (image modèle) est préenregistrée dans notre programme en C, le drone fait l'acquisition d'images par flux vidéo et les compares afin de garder une seule image qui lui indiquera le déplacement à effectuer.

**Algorithme :**

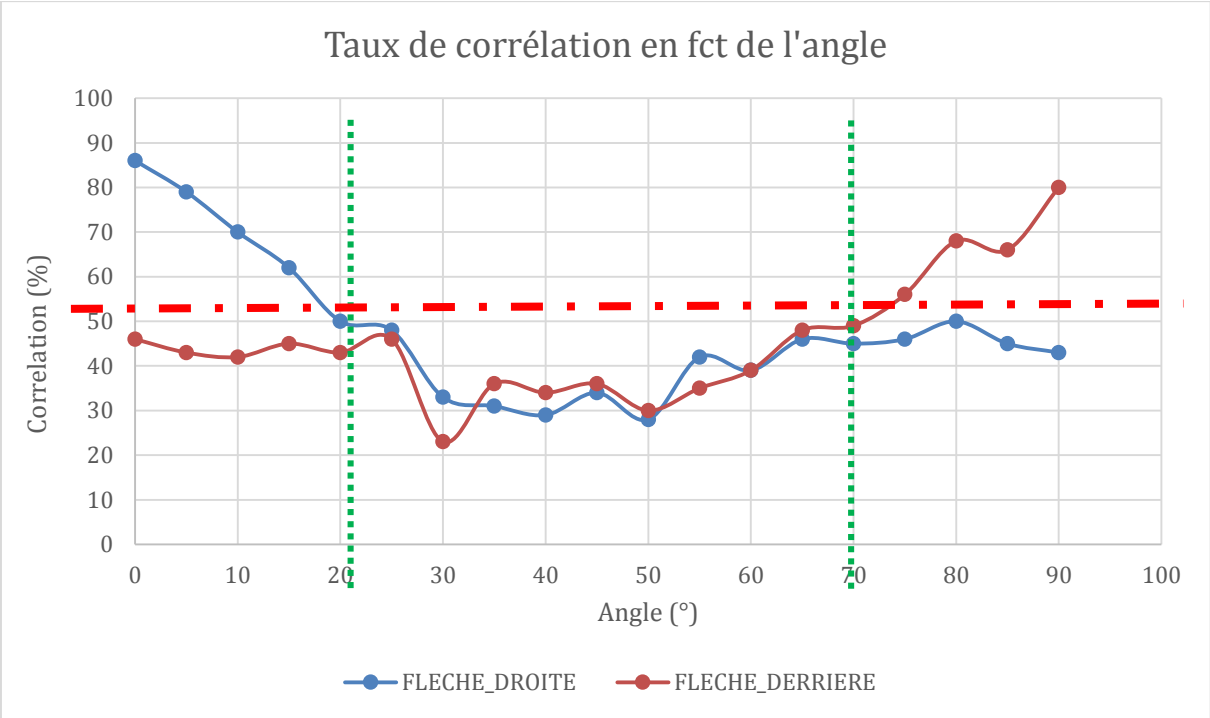


## 1. Etude préliminaire

Explication de l'étude du seuil de corrélation :

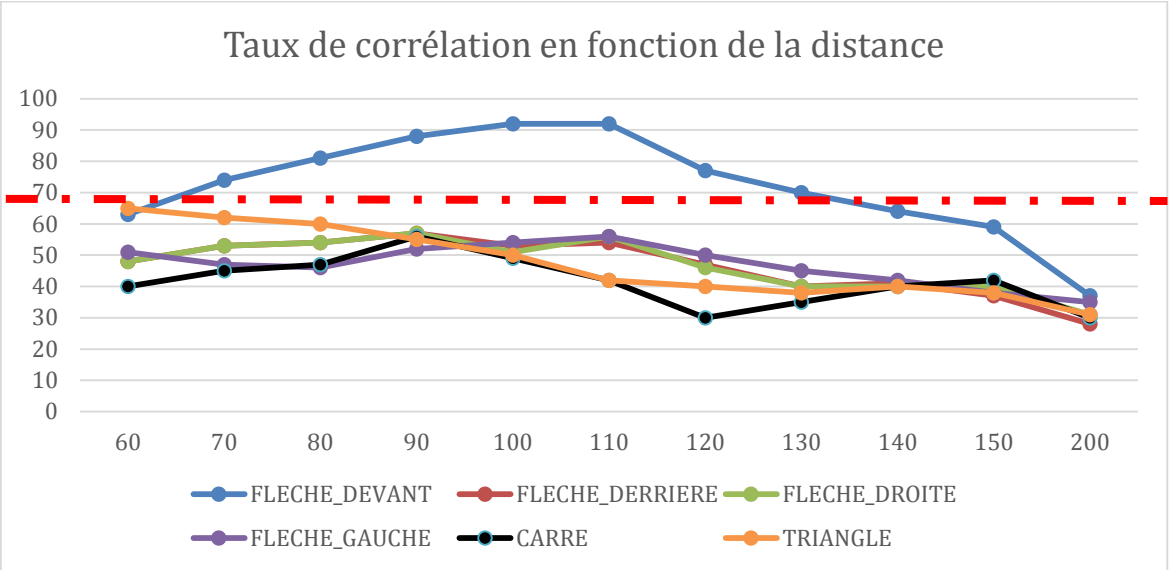
Nous avons fait l'étude du taux de corrélation en fonction de l'angle, le taux de corrélation en fonction de la distance et avons superposé nos résultats :

- Etude flèche à l'échelle 1 :1 : taux de corrélation en fonction de l'angle



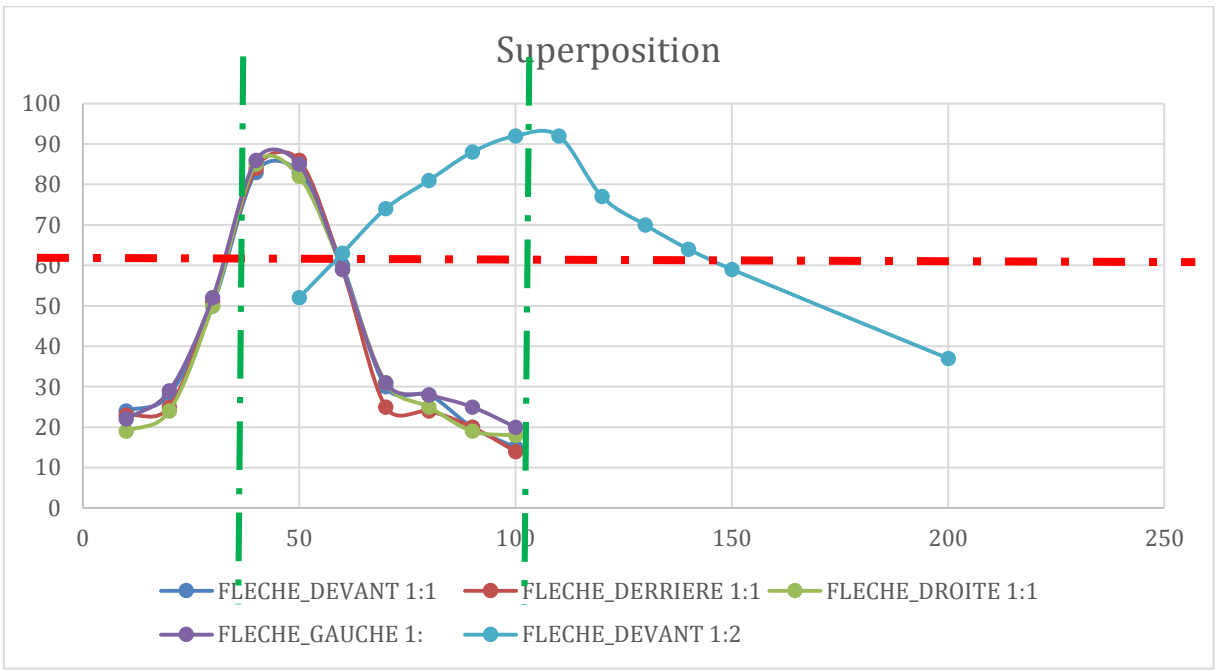

Nous avons fixé un seuil à 55% (endroit ou l'on obtient la meilleure corrélation), puis avons relevé en faisant varier l'angle de 5 degrés à chaque fois la valeur de corrélation obtenue. Nous obtenons donc deux courbes correspondantes aux flèches de droite et arrière. On remarque un palier compris entre 20° et 70° qui montre que les flèches sont bien reconnues et sont concourantes.

- Etude flèche à l'échelle 1 :2 : taux de corrélation en fonction de la distance (flèches, rectangle, triangle) :




Ce graphique nous montre le taux de ressemblance des flèches en fonctions de la distance. Nous avons fait nos essais avec des flèches, mais aussi un rectangle simple, un triangle et un carré afin d'être le plus précis possible dans la corrélation de nos flèches.

- Graphique de superposition :




En superposant nos résultats, on remarque que les courbes représentant chaque flèche se chevauchent ayant donc la même courbe. Nous avons donc superposé avec ces 3 courbes représentant les flèches à l'échelle 1 :1 la courbe bleue (représentant la flèche de devant à l'échelle 1 :2) afin de trouver le seuil que nous

allions appliquer à notre programme (intersection entre courbe violette et la courbe bleue).

Ce graphique nous a donc permis de trouver un seuil d'environ 55% compris entre un angle de 30° à 100°.

## 2. Explication du programme

### i. Acquisition vidéo

Le drone va acquérir des images à l'aide de sa caméra du dessous. Pour récupérer le flux vidéo de cette caméra nous allons utiliser :

- CvCapture* capture = cvCreateCameraCapture(CV_CAP_ANY);

Cette fonction va créer un flux vidéo « capture ». Son argument « CV_CAP_ANY » signifie qu'il va détecter automatiquement la caméra du drone.

- IplImage *src=cvCreateImage(cvSize(640,480), 8, 3);

Cette fonction va créer une image « src » de taille 640*480 pixels, de profondeur 8 bits et de canal 3 où qui contiendra toutes les images du flux vidéo capture.

- src = cvRetrieveFrame( capture );

Cette fonction enregistre les images du flux vidéo « capture » dans « src »

### ii. Chargement d'images

Chargement des 4 images avec cvLoadImage : les images sont pré-enregistrées

```
droite = cvLoadImage("fleche_droite.png",1);
gauche = cvLoadImage("fleche_gauche.png",1);
haut = cvLoadImage("fleche_devant.png",1);
```

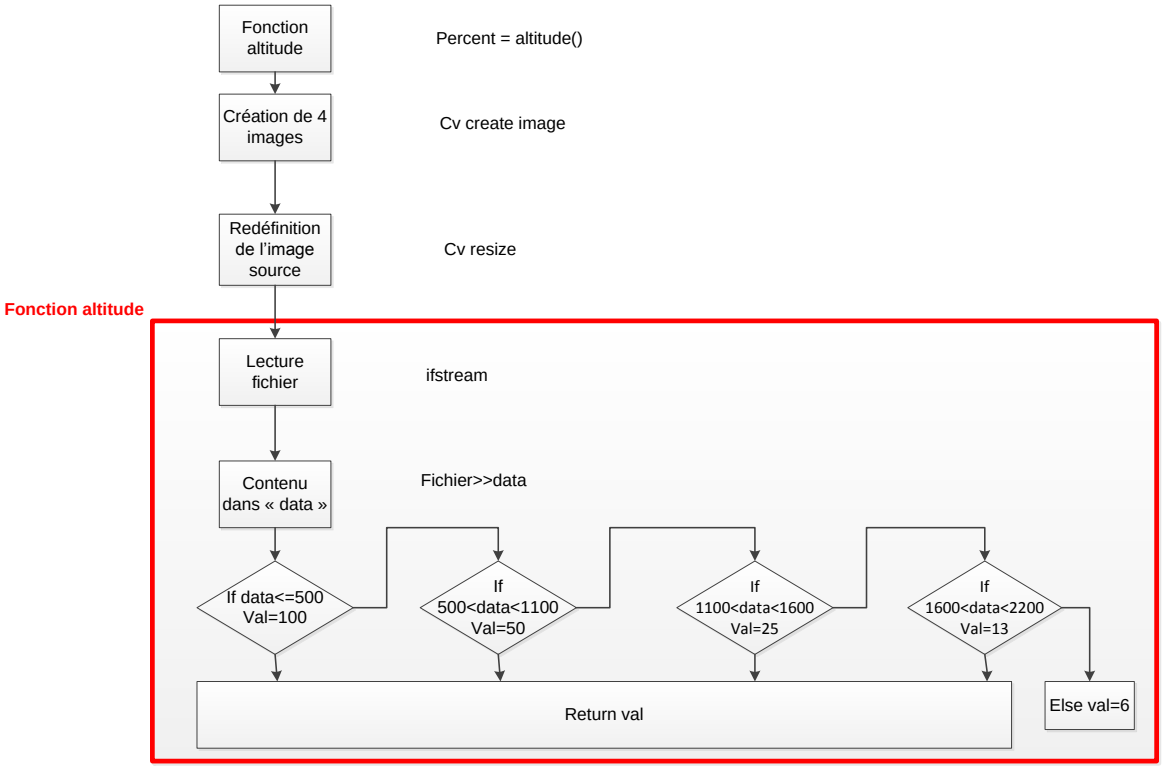
### iii. Sur échantillonnage

On utilise la méthode de sur échantillonnage ; il y a deux méthodes qui peuvent être appliquées dont on a retenu seulement la première décrite.

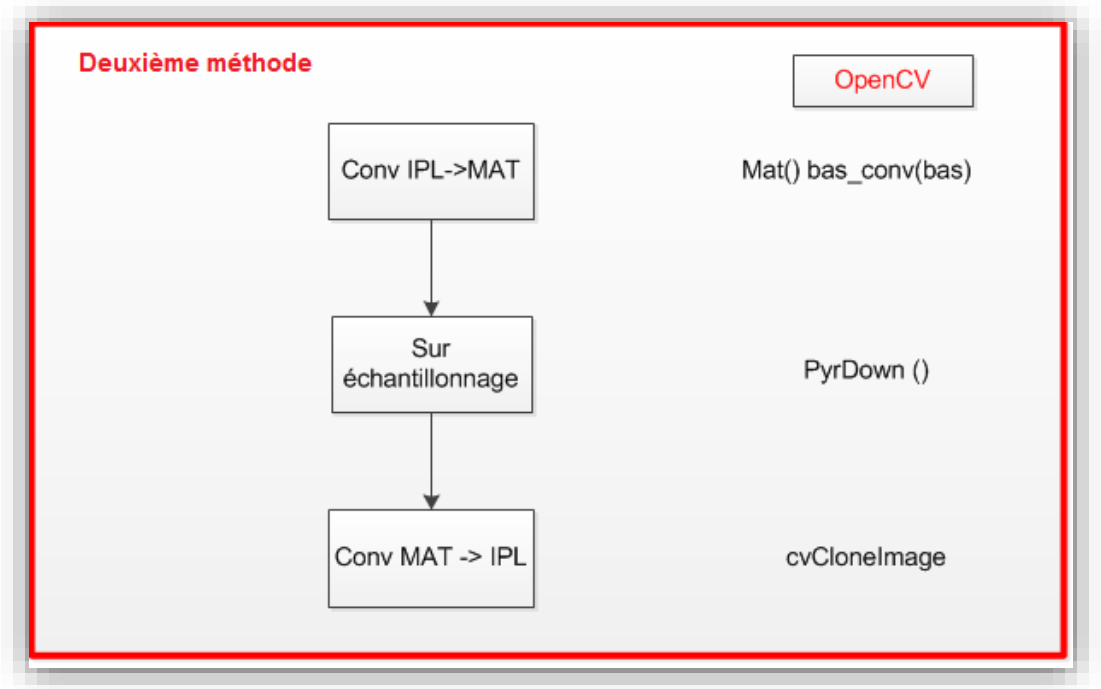
La première méthode utilise une variable qui va contenir le pourcentage de redimensionnement de l'image en fonction de l'altitude.

La deuxième méthode (plus rapide) selon la technique de la pyramide utilise le pyrDown mais nous ne pouvons pas l'utiliser car on ne peut pas faire deux pyrDown d'affilé pour le redimensionnement d'image donc cette méthode n'a pas été retenu.

Voici la première méthode : diagramme explicatif




Voici la deuxième méthode : diagramme explicatif




On Convertit l'IplImage en Matrice. On crée des matrices temporaires dans lesquelles le sur-échantillonnage va être stocké. On déclare la destination IplImage Object avec la taille, la profondeur et le Channel. Le pyrDown divise par deux la taille de la flèche. On convertie l'image Mat en IplImage grâce à cvCloneImage en castant.

### iv. Création des 4 images

On va créer nos quatre images de référence : par exemple pour la flèche d'indication « devant » :

- IplImage *templ = cvLoadImage("fleche_devant.png",1);

Cette fonction va charger l'image fleche_devant se trouvant dans le répertoire courant du projet. On enregistre cette image dans « templ » qui sera notre image template c'est-à-dire notre image modèle (de référence).

Ensuite nous allons créer 4 images temporaires qui contiendront les images résultats de chaque corrélation : par exemple pour la flèche d'indication « devant » :

```
IplImage CreationImageFTMP(IplImage *src,IplImage *templ)
//src=image camera
//templ=image de référence
{
//définition de la taille(largeur, hauteur) de l'image ftmp
   int iwidth = src->width - templ->width + 1;
   int iheight = src->height - templ->height + 1;

//Creer un pointeur d'image ftmp de type IplImage et de taille iwidth et iheight
   IplImage *ftmp = cvCreateImage(cvSize(iwidth,iheight),IPL_DEPTH_32F,1);
        return *ftmp;
}
```

Cette fonction que nous avons créée nous même fait :

- Calcule la taille de l'image temporaire en fonction de l'image source « src » et template « templ »
- Créer l'image ftmp de taille défini ci-dessus, de profondeur 32 bits et de canal 1

### v. Création d'un cadre de détection

Nous utiliserons ensuite une fonction permettant de savoir si la flèche a été ou non détecté par le drone. Cette fonction permet de créer un encadré rouge qui suit la flèche détectée : par exemple, pour la flèche « devant » :

- Création des points qui définiront cet encadré rouge

```
CvPoint cadre_pt1 = cadre_pt0(src,templ_1);
CvPoint cadre_pt2 = cadre_ptbis(src,templ_1,cadre_pt1);
```

- Création du rectangle

```
cvRectangle(src, max_loc,max_loc2, cvScalar(0,0,255));
```

Cette fonction trace un rectangle sur l'image source « src », de dimension max_loc et max_loc2 (qui sont en fait 2 points qui définisse les coins du rectangle) et cvScalar qui permet de rejoindre ces 2 points.

## vi. Fonctions de corrélation

- Corrélation entre l'image source « src » et l'image de référence « templ_1 »

```
cvMatchTemplate( src, templ, &ftmp, CV_TM_CCOEFF_NORMED);
```

Cette fonction permet de connaitre le taux de corrélation entre 2 images grâce à cvMatchTemplate. Les arguments sont : l'image source « src », l'image de référence « templ », l'image de résultat ftmp et le type de corrélation : CT_TM_CCOEFF_NORMED.

**method=CV_TM_CCORR_NORMED**

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

**Explication :** cette fonction va déplacer l'image template sur l'image source verticalement (de bas en haut) et horizontalement (de droite à gauche).

Exemple du tutoriel d'OpenCV pour cette fonction : la photo de la tête du chien va être glissée de gauche à droite et de bas en bas pour connaitre le taux le plus ressemblant.

- Nous utilisons ensuite une fonction permettant de trouver l'endroit où la corrélation est la plus élevée.

```
cvMinMaxLoc(&ftmp, &min_val, &max_val, &min_loc, &max_loc);
```

Cette fonction a pour argument : l'image résultat « ftmp », la valeur minimale de corrélation « min_val », la valeur maximale de corrélation « max_val », la position du min_val « min_loc » et la position du max_val « max_loc ».

### vii. Fonctions de comparaison

Une fois les taux de corrélation obtenus, on veut savoir laquelle des 4 flèches (avancer, reculer, droite ou gauche) a été reconnu. Pour cela nous avons fait une fonction :

```
int tab[4]={max_val_100,max_val_100_2,max_val_100_3,max_val_100_4};

        //Initialisation de 3 variables
        int max=0; //Va contenir le plus grand élément du tableau

        //Boucle qui permet de connaitre la plus grande valeur dans le tableau
                for (int i=0 ; i < 4; i++)
                {
                        if (tab[i]>max)
                        {
                                max=tab[i];
                        }
                }
```

Ici, nous rangeons les 4 valeurs de corrélation dans un tableau. Nous savons que la première case du tableau « max_val_100 » correspond à la flèche « avancer ». Donc nous vérifions laquelle des valeurs (après la corrélation) du tableau est la plus grande.

### viii. Affichage de la direction de la flèche et du taux de corrélation

Pour tester notre programme avec notre webcam, nous avons défini des conditions pour vérifier le bon fonctionnement de notre programme :
Si la valeur de corrélation de la flèche « avancer » correspond au taux de corrélation maximum et si le taux de corrélation est supérieur à 55% alors, afficher dans la console : « FLECHE AVANCER à x% » où x correspond au taux de corrélation actuel.

```
if((max == max_val_100) && (max_val_100 > SEUIL))
        cout << "=== FLECHE AVANCER \x85 "<< max_val_100 <<"===" << endl;
```

## C. Détection par forme

Voici un schéma (figure 1) récapitulatif de notre programme, l'image va être traitée de 2 façons différentes puis va nous donner l'orientation et le sens à suivre pour la flèche.

Les différentes images des flèches présentes sur le diagramme ci-dessous montrent ce que le programme effectue à chaque nouvelle étape.

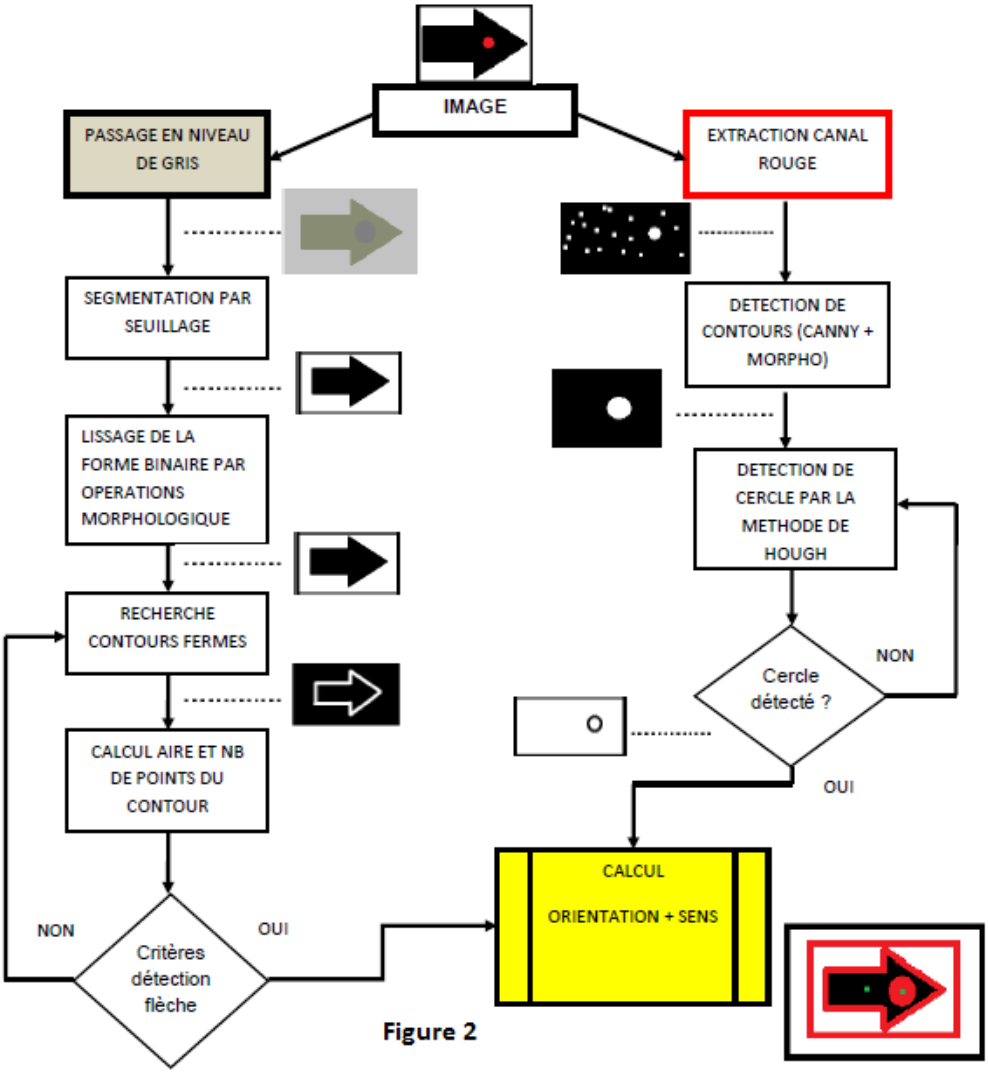



**Figure 2**

Cette méthode consiste à détecter des formes géométriques qui constituent et définisse l'objet que l'administrateur a choisi d'identifier.

Pour notre part nous avons choisi de détecter la forme d'une flèche :

figure. 2

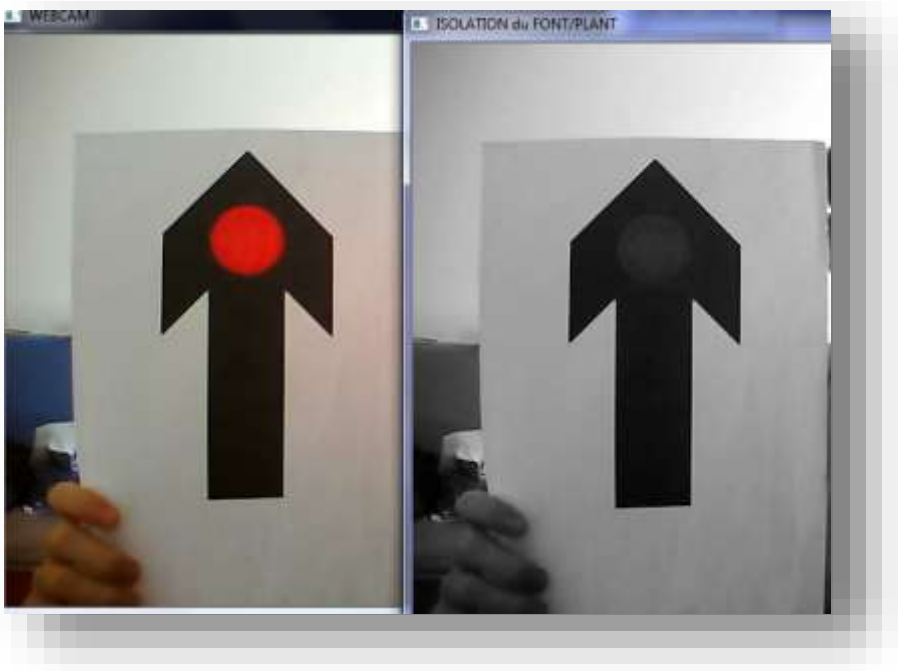
Notre flèche a une forme géométrique bien précise.
Elle contient :

- 9 segments
- 9 angles
- une aire selon sa prise de vue
- un Barycentre (centre d'inertie)

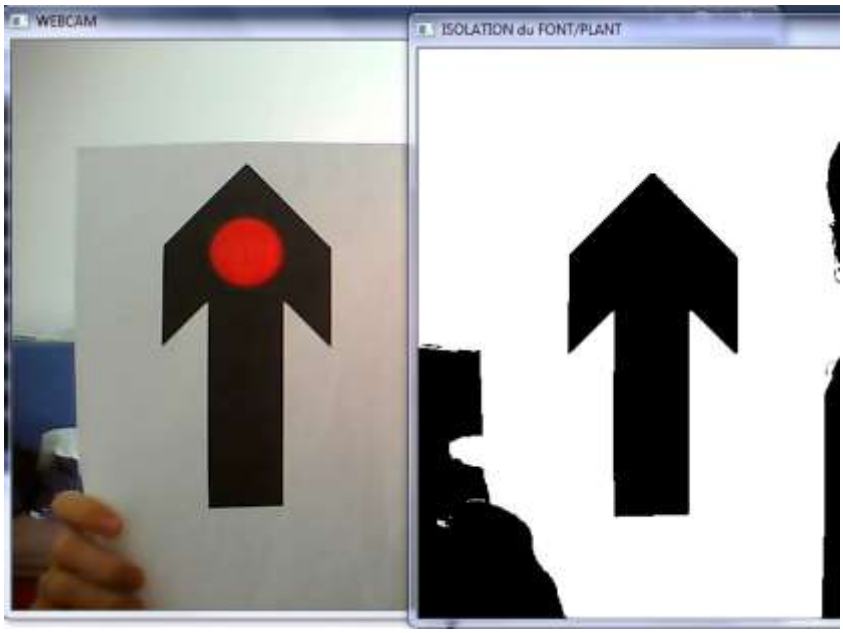
## Détection de la flèche :

Nous allons donc dans un premier temps garder seulement les contours de notre image. Pour cela nous avons besoin de lisser l'image ainsi que d'isoler les différents contours de notre image par le biais de filtre.

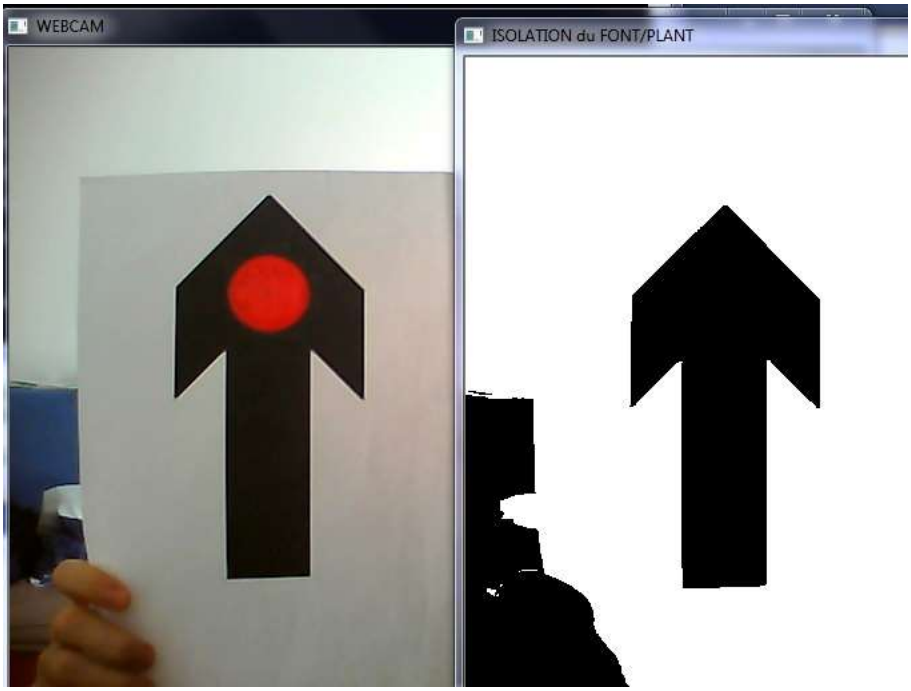
1ère filtre : Conversion de notre image en niveau de gris



2ème filtre : Segmentation de l'image par seuil binaire




3ème filtre : Filtre morphologique (met plus en évidence nos contours)



Notre image est utilisable pour rechercher des formes précises. Nous allons donc enregistrer chaque point décrivant des contours fermés de nos objets.

Ce sont des points qui ont été enregistré et les contours ne sont pas très précis, ils décrivent une forme par un nuage de points. Nous allons donc déduire le nombre de point significatif pour chaque forme par l'intermédiaire d'un algorithme qui effectue un contour en pointillé. Ce qui nous donne une forme géométrique soit plusieurs polygones.

Nous ne voulons détecter que les flèches, c'est ainsi que nous allons monter notre algorithme de sélection pour ne garder que des polygones à 9 cotés ; c'est à dire nos ennéagone

Maintenant il se peut que des petits ennéagones surviennent dû à la lumière. Pour éviter un maximum d'erreur nous partons du principe que notre flèche est discernable à l'œil donc qu'elle a une aire suffisamment grande. Nous avons déterminé aussi une sélection de l'aire de l'ennéagone défini par l'administrateur.

Nous avons donc bien notre flèche de détecté. Or ces informations ne sont pas suffisantes, nous ne savons pas si la flèche indique le haut ou le bas ou la gauche ou la droite ou une direction particulière.

Pour ce faire nous allons donc chercher 2 points significatifs de notre flèche.

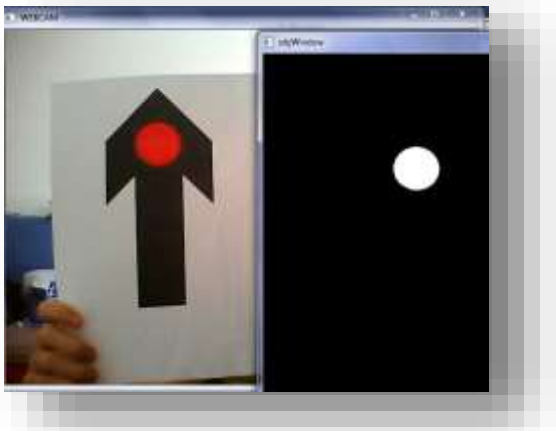
## Création de deux points :

### Le premier point :

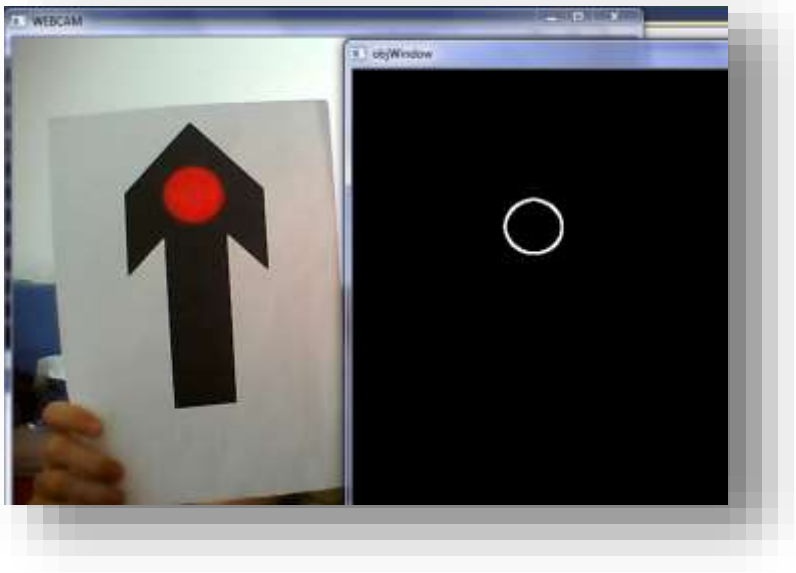
Nous avons décidé de prendre le centre de la flèche c'est à dire le barycentre de cette flèche. Car un des points retenu par l'approximation n'aurai pas été un repaire visuel pour nous (ne connaissant pas l'ordre dans lequel les points de la flèche ont été enregistré). Pour ce faire nous avons calculé le moment d'inertie des points significatifs de la flèche.

### Le deuxième point :

Nous avons décidé de rechercher un cercle mais surtout son centre. Il nous fallait un nouveau paramètre de distinction des formes : le plus rapide qui nous est apparu, fut la détection par couleur. Nous avons isolé notre couleur rouge et obtenu la forme du cercle.



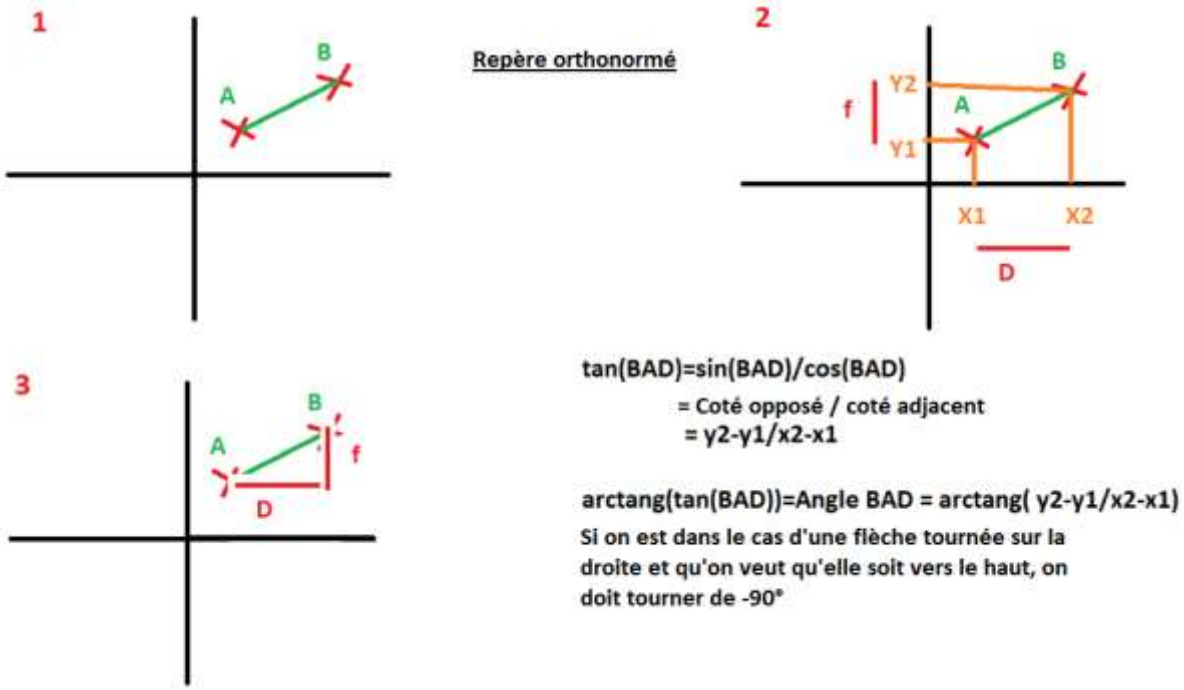
Il nous a fallu le détecter : cette forme géométrique est détecter par l'intermédiaire de la transformé de Hough. Pour cela, nous sommes passés par un filtre de canny et une dilation :




Le cercle est ainsi détecté, ses points sont enregistrés, nous avons donc notre 2ème point.

## Calcul de l'angle :

Pour cette 3ème partie nous avons cherché une solution qui avec 2 points nous permettrais d'avoir l'angle entre ces 2 points.



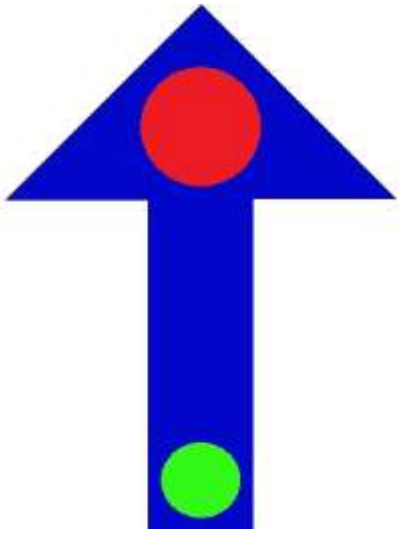

Nous avons notre angle avec notre flèche et notre cercle détecté.

En plus de l’angle défini, nous voulions savoir de quel angle on doit tourner pour voir cette flèche droite.

## D. Détection par la couleur

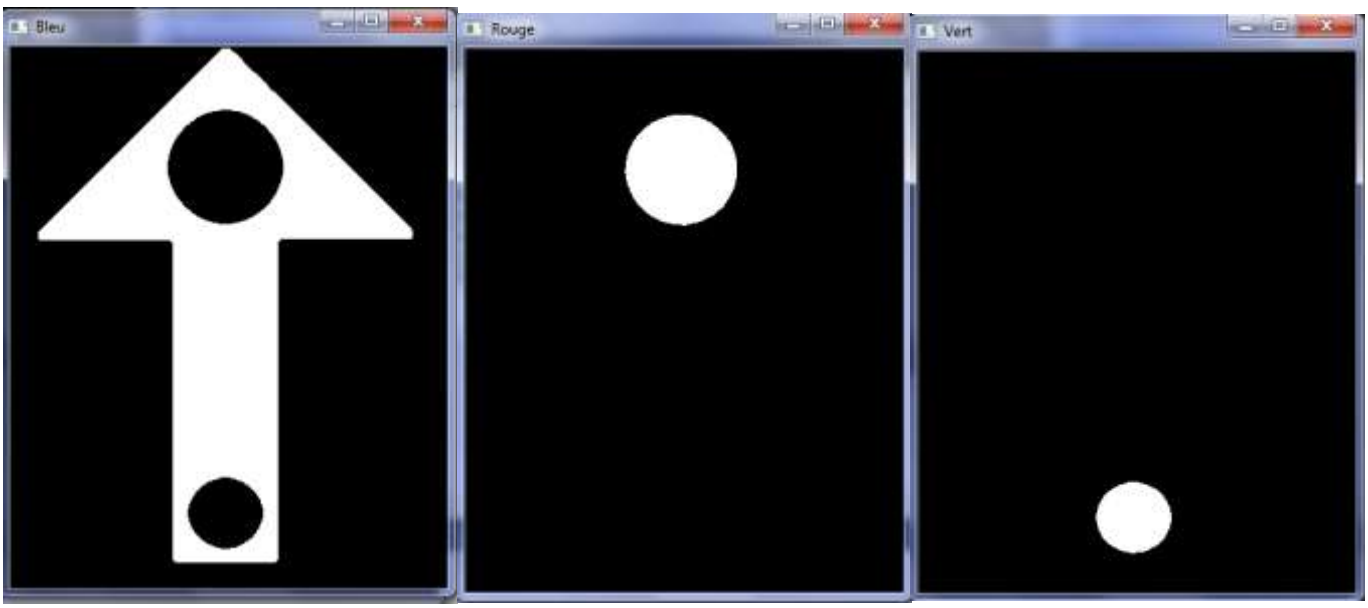
Cette méthode consiste à isoler certaines couleurs ciblées par l'utilisateur de telle sorte à détecter les couleurs qui présentent un intérêt pour cibler des objets particulier dans une image.

Dans notre cas nous voulons détecter cette flèche :



Dans un premier temps, nous voulons isoler les 3 couleurs : rouge, vert et bleu grâce à Inrange (qui permet de « voir » une image dans une certaine gamme de couleur) :

<u>Isolation du bleu, rouge et vert :</u>

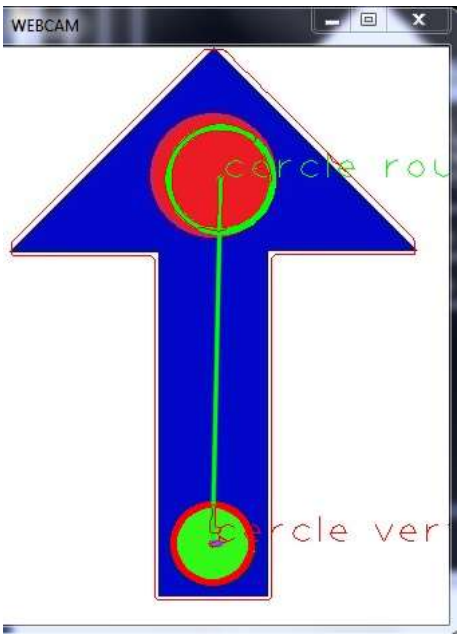


Dans un 2ème temps nous voulions connaitre la position du rouge en additionnant l'image du filtre vert et bleu, et la position du vert en additionnant l'image du filtre bleu et vert.

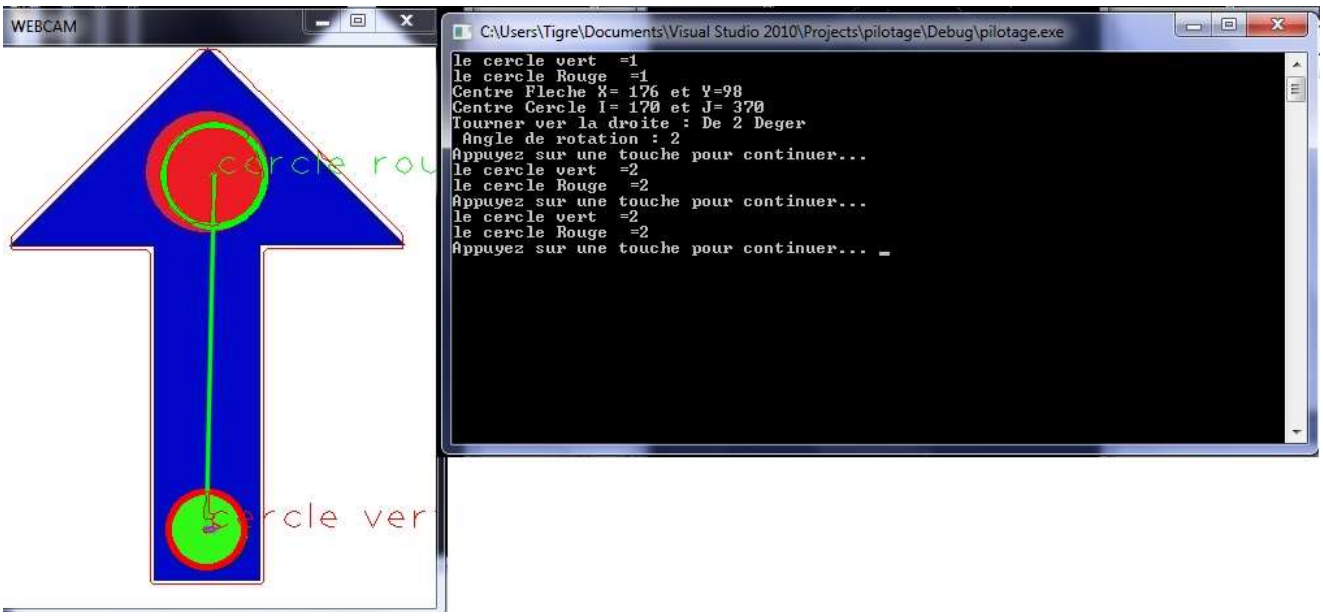


Suite à ça, nous avons additionné ces 2 images pour donner notre flèche pleine :

Nous pouvons maintenant détecter le contour de la flèche et détecter le centre de nos 2 cercles. Pour obtenir :




En reprenant l'algorithme du calcul de l'angle nous pouvons maintenant connaitre l'angle de notre flèche et le renvoyer au drone :




Méthode beaucoup plus rapide, mais qui peut comporter des inconvénients selon la lumière à laquelle l'image est soumise.

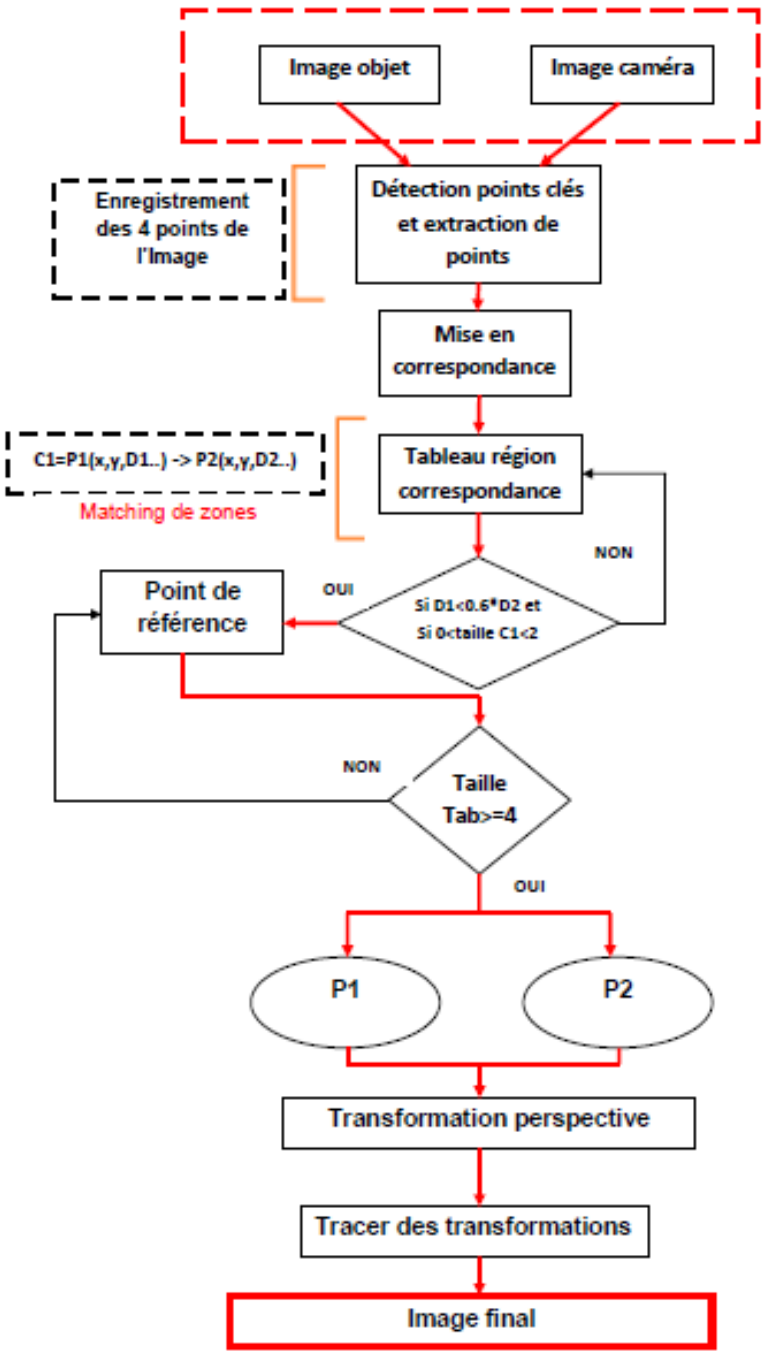
## E. Détection par points d'intérêts

Cette méthode consiste à détecter des points locaux dans des régions qui semblent pertinents sur une image de référence (généralement des angles) et ainsi faire la reconnaissance point à point.

Méthode assez rapide car elle ne compare que quelques points d'une image.

<u>Voici un schéma récapitulatif de notre programme :</u>

Notre programme utilisant la méthode des points d'intérêts se prénomme surf.



Initialisation :

Enregistrement de l'image de référence dans une matrice Image (la flèche) en image de gris.

- On enregistre 4 points de l'image dans un tableau tel que :
    - Point en 0 (0 ;0)
    - Point 1 (Nb de colonne de l'image, 0)
    - Point 2 (Nb de colonne de l'image, Nb ligne de l'image)
    - Point 3 (0, Nb de ligne de l'image)

Application d'une fonction :

- Détecte les points clés de l'image de référence
- Enregistrement des zones (les angles et bordures)

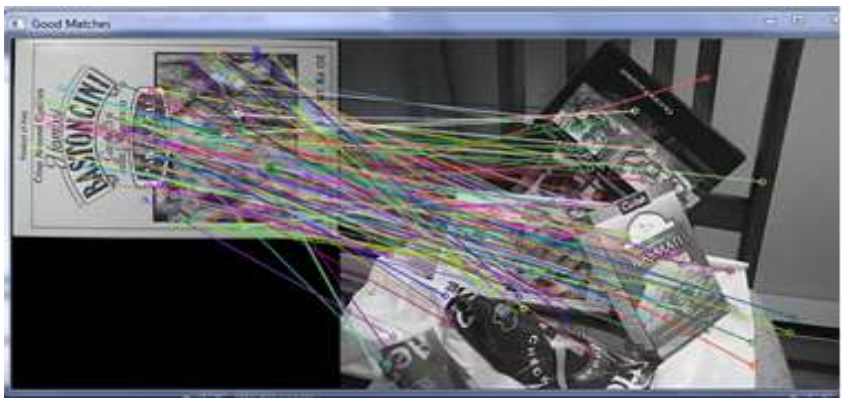
Exemple : sur cette photo les régions ciblées ne sont que des angles et des bords des objets.

Toutes les zones ciblées doivent maintenant être comparées pour dire si oui ou non elles correspondent à ceux de l'image de référence.
Nous utilisons pour cela la fonction KNN (k algorithme-plus proches voisins) qui est une méthode utilisé pour la classification et la régression.

KNN calcule essentiellement la distance entre une zone de référence et la zone de l'image à comparer il recherche la région qui pourrait ressembler à une des régions prise en référence

- Trouve les 2 meilleures correspondances pour chaque zone
- Donc par pixel il compare les points des zones de l'image de référence et en sort pour chacun une comparaison 2 meilleurs zones référence.

Exemple de résultat :




La correspondance ne veut pas dire que c'est exactement la même zone, il faut donc les triller et mettre des facteurs de sélection pour éviter que toute zone qui ressemble plus ou moins à une autre soit choisie.
Nous avons donc procédé à un algorithme de filtrage :

Algorithme de filtrage :

Pour chacune des meilleures correspondances on va déterminer si on les gardes ou pas dans un tableau nommer « good » :

- Si la distance de la région de référence est inférieur à 0.6* (région de l'image sélectionnée) et si $0 < Taille \leq 2$ alors je garde la zone comme bonne zone de référence.

Fonction de dessin de correspondance de point entre les points jugé bon et leur référence : Algo2 :

- Si la taille du tableau good est supérieur ou égale à 4 (il ya au moins 4 points de référence) alors j'enregistre ses coordonnées dans 2 autres tableaux (un pour l'image de référence et un pour l'image saisie).
- Une fois les 2 tableaux rempli on fait faire le calcul de la meilleure transformation de perspective entre les deux plans (les 2 tableaux qu'on a rempli).
- J'enregistre ces résultats dans une matrice.
- Je fais la transformée en perspective des 4 points de l'image enregistré au début du programme et la matrice que je viens d'enregistrer.
- Ensuite je trace les lignes de la transformée sur l'image de la caméra.

Autre explication :

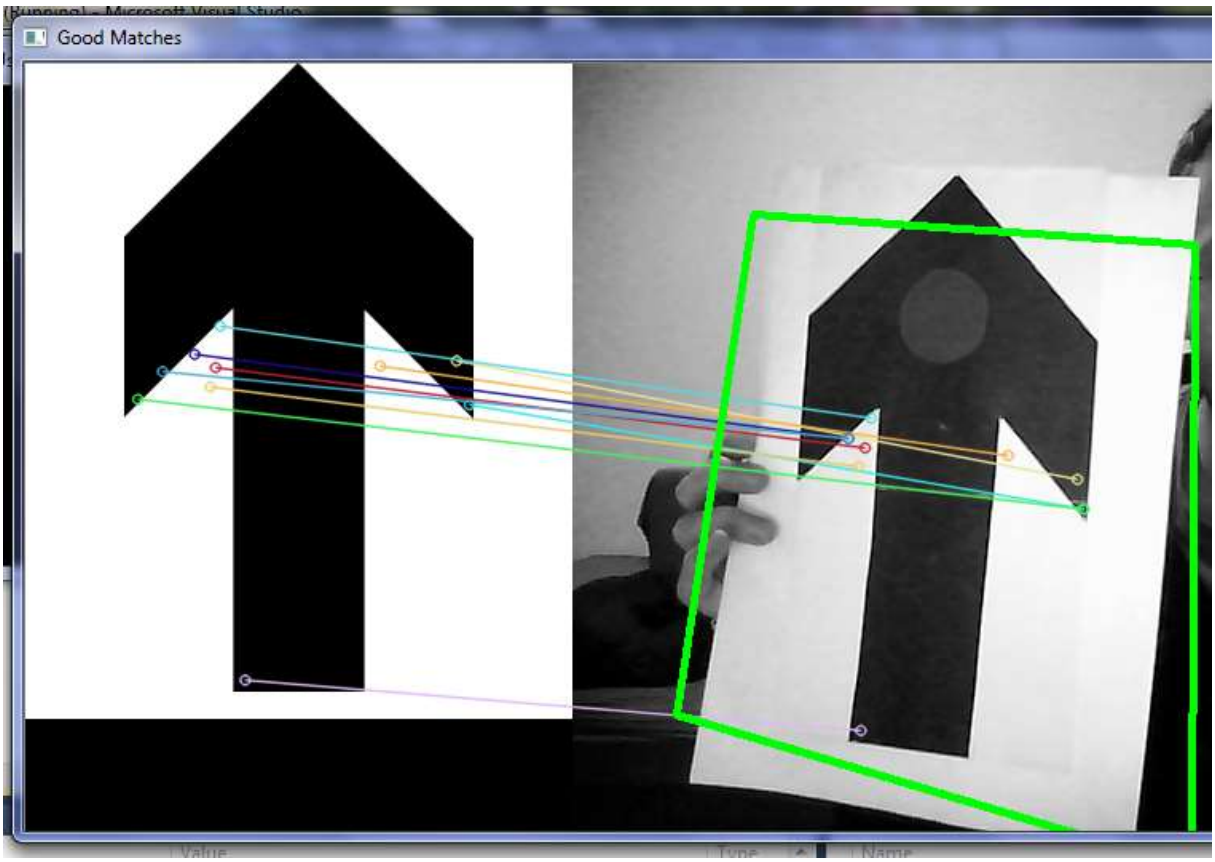
Prenez une image de l'objet que vous voulez détecter et extrayez les descripteurs SURF (Speed Up Robust Features). Il est important pour que cette image contienne seulement l'objet et soit à l'abri de toute lumière crue.

• Maintenant, faites la même chose pour chaque image provenant de votre appareil photo.
• Utilisez la stratégie d'adaptation pour correspondre à des descripteurs de chaque image avec les descripteurs de l'objet et trouver les «bons matchs ».
• Créez une fenêtre avec l'image de l'objet sur un côté et le jeu vidéo de l'autre côté. Tirer de bonnes correspondances entre les deux.

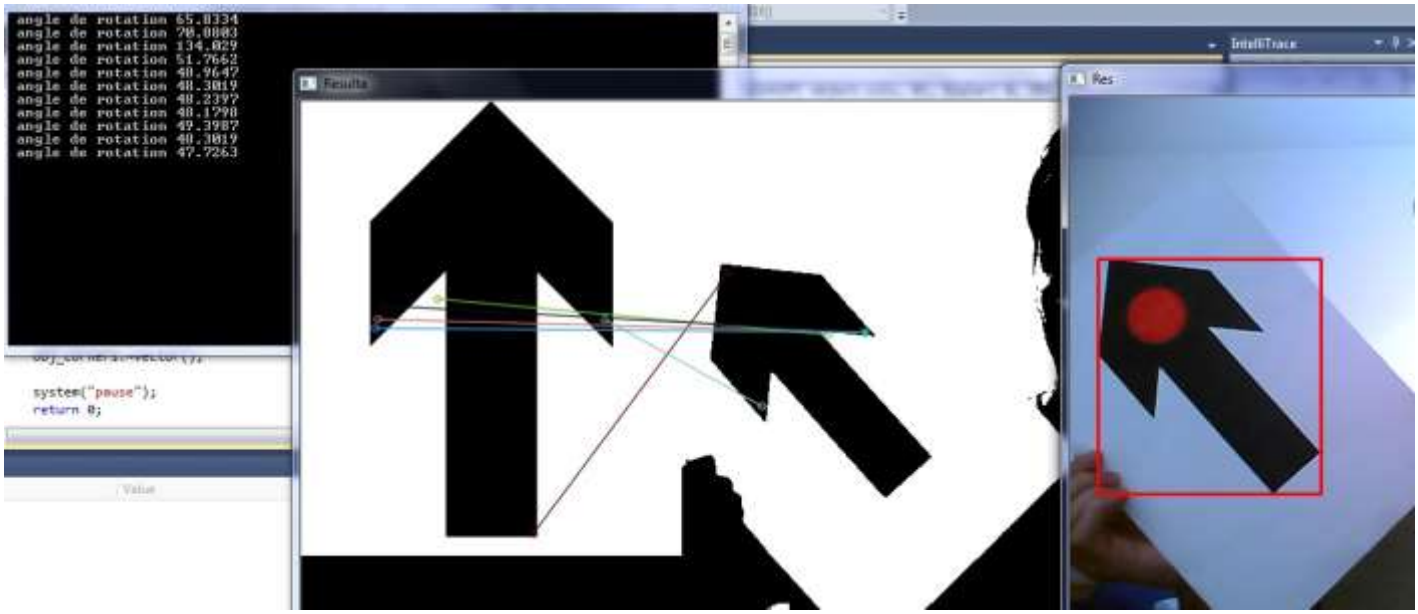
• Pour obtenir un cadre de sélection autour d'un objet détecté, en utilisant les bons matches, trouvez une homographie qui transforme les points de l'image de l'objet à l'image vidéo.

Par utilisation de cette homographie, transformer les 4 coins de l'image de l'objet. Considérer ces quatre points transformés comme les sommets et tracer une zone dans l'image vidéo.

La partie la plus importante est la stratégie d’adaptation : SURF (comme son nom l’indique) est assez rapide pour faire tout cela pour chaque image provenant de votre appareil photo. Vous pouvez obtenir quelque chose comme ceci :



Nous avons modifié le programme pour que ce dernier nous calcul l’angle de notre flèche :



# IV. Commande du drone

## A. Introduction

Nous avons dû apprendre le langage de programmation du drone dans le but de pouvoir le contrôler. Ce langage est le Node JS. Pourquoi avoir utilisé le Node JS ?

- Simplicité pour le développement
- Adapté au temps réel
- Une communauté plus grande et plus réactive
- Nouvelle technologie

## B. Manipulation du drone

Dans un premier temps nous nous sommes documentés sur la manipulation du drone :

- Avancer
- Reculer
- Tournoi à droite
- Tournoi à gauche
- Translater à droite
- Translater à gauche
- Monter
- Descendre
- Faire des figures
- Atterrir
- Décoller

**Ligne en node JS :**

```
//Bibliothèque ar drone
var arDrone = require('ar-drone');
//On crée un client (Drone)
var client = arDrone.createClient();

//Vitesse du drone comprise entre 0 et 1
client.up(0.5);                 //Monter
client.down(0.5);               // Descendre
client.front(0.5);              //Avancer
client.back(0.5);               //Reculer
client.takeoff();               // Décoller
client.land();                  // Atterrir
client.right(0.5);              //Translater à droite
client.left(0.5);               //Translater à gauche
client.counterClockwise(0.5);   //Tournoi à droite
client.clockwise(0.5);          //Tournoi à gauche
client.stop();                  // S'arrêter
client.getVideoStream();        //Obtenir la vidéo du drone
client.getPngStream();          // Obtenir une image PNG du drone
```

## C. L'interface

Interface de test de contrôle du drone :

Pour sélectionner la caméra du dessous, il faut rajouter une ligne de code. Cela va configurer notre drone :
Il faut aller dans la bibliothèque « Ar-Drone », puis ouvrir « index.js » et rajouter la ligne :

```
//Bottom camera
client.config('video:video_channel', 3);
```

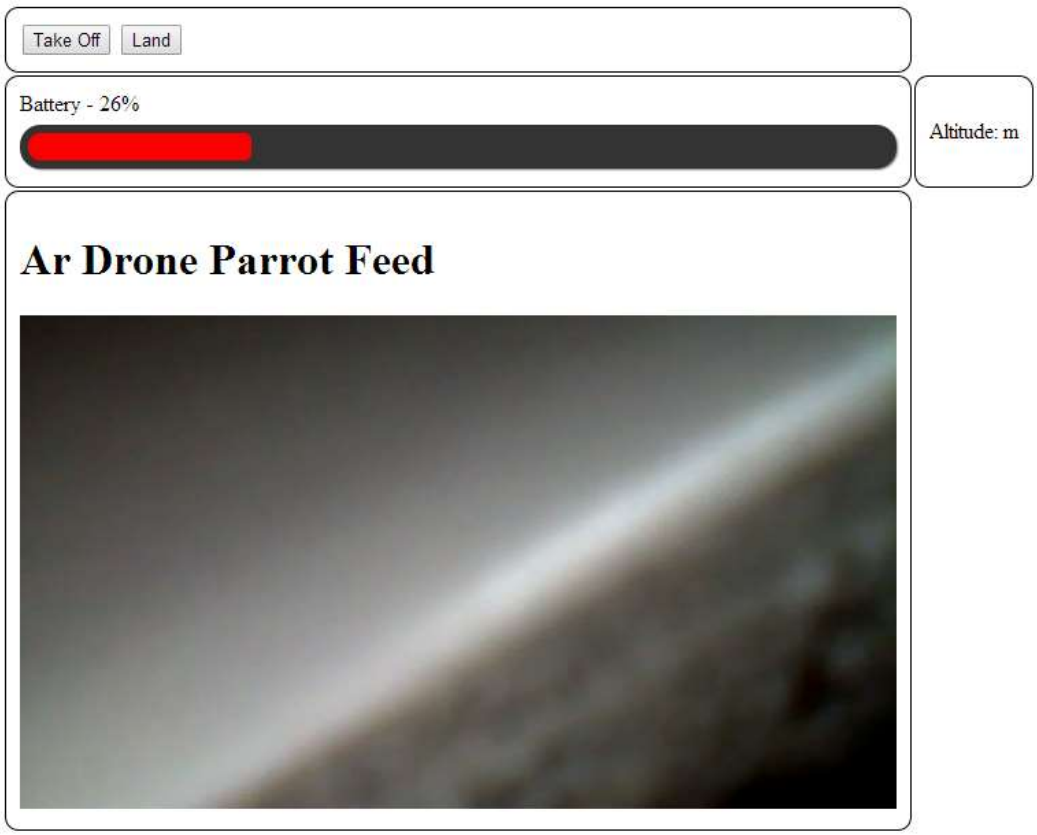
Il faut la rajouter dans la fonction qui s'occupe du client (c'est-à-dire notre drone) :

```
exports.createClient = function(options) {
  var client = new arDrone.Client(options);

  //Select bottom camera
  client.config('video:video_channel', 3);

  client.resume();
  return client;
};
```

Nous pouvons voir la caméra ventrale du drone qui renvoie le flux vidéo (30 images par seconde), le niveau de chargement de la batterie, l'altitude du drone lors du vol, ainsi que deux boutons de mise en marche et d'arrêt.



# V. Conclusion

## A. Résumé

Au commencement de notre projet nous avons cherché des solutions pour résoudre notre problématique qui était de permettre à un Ar Drone de s'auto-diriger par indication de flèches au sol.

Une fois notre cahier des charges posé, nous avons commencé à travailler sur le traitement d'images en utilisant le langage C/C++ et la bibliothèque openCv. Tout en faisant des tests avec notre programme, nous avons du déterminer un seuil de détection en conformité avec la distance et l'angle afin que nos images soit détectées et reconnues avec le moins d'erreurs possible.

Après la partie traitement d'images, nous avons appris le langage de programmation du drone pour pouvoir le contrôler (Node JavaScript).

Enfin nous avons étudié d'autres méthodes pour pouvoir améliorer le temps de détection et de reconnaissance de nos flèches pour que notre système final se rapproche au plus de ce que l'on souhaitait faire.

## B. Projection dans l'avenir/Amélioration

Concernant l'Ar Drone de Parrot sur lequel notre projet était basé, des améliorations peuvent être apportées, que ce soit sur l'asservissement a proprement dit du drone afin que le moins de données possible soit perdues et qu'il est un déplacement quasi parfait, soit au niveau du développement des possibilités du drone.

D'autres façons de contrôler le drone peuvent être conduites en projet comme l'utilisation d'un capteur en faisant déplacer le drone avec la main, ou encore contrôler le drone par la pensée en utilisant un casque et les ondes.

Dans l'avenir et dans le cas où le drone pourrait être contrôlé à la fois par la main et par la pensée, une amélioration à distance en suivant un homme en autonomie complète pourrait être l'avenir du robot Ar drone afin qu'il soit mis en circulation dans la vie de tous les jours pour des applications dans des usines ou dans la sécurité dans différents domaines.

# VI. Bilan

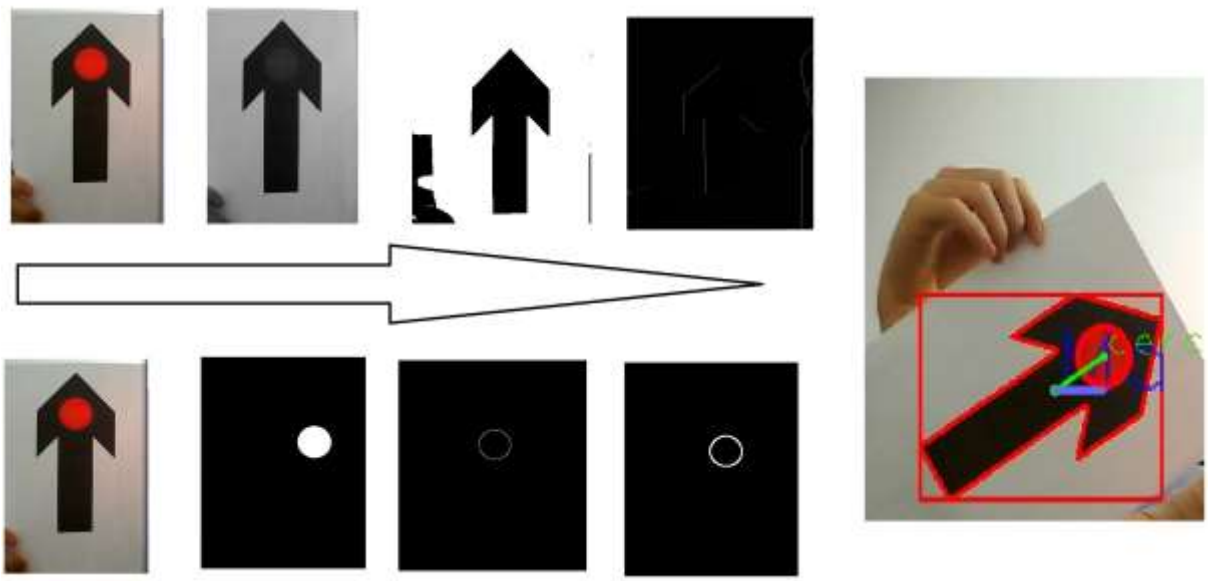
Concernant la méthode de corrélation :

Notre programme en C++ fonctionne correctement, il corrèle nos 3 images mais la méthode prend beaucoup trop de temps par rapport au temps de réaction du contrôle du drone. Cette méthode demande beaucoup de ressource à l'ordinateur, ce qui rend le traitement lent, et pas vraiment adapté à la détection en temps réel.

+ PHOTO IMPRI CORRE

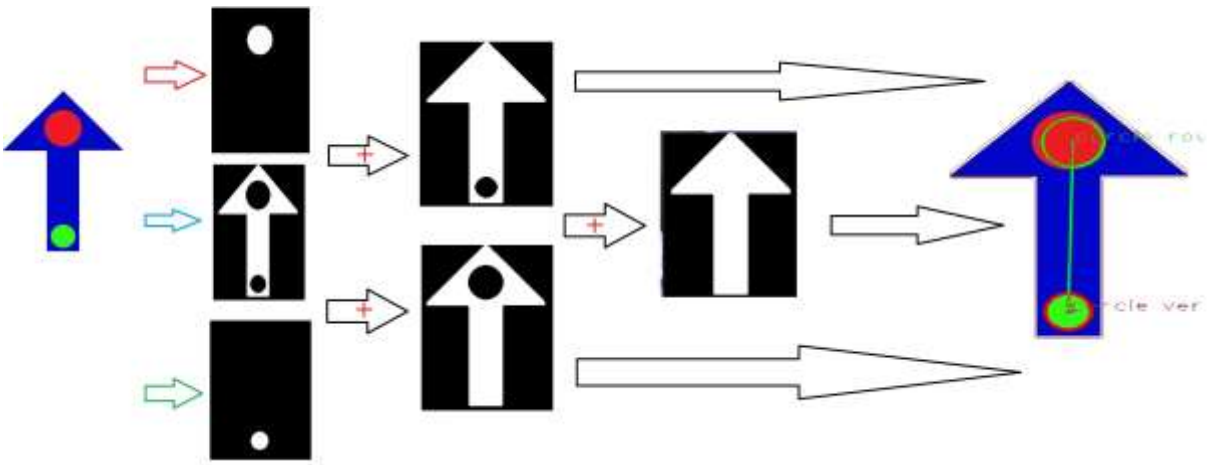
Concernant la méthode des formes :

Notre programme en C++ fonctionne correctement. La forme de la flèche et le centre du cercle sont bien détectées. Le calcul s'effectue rapidement. Les principaux problèmes à cette méthode sont la luminosité et la détection d'autres formes qui peuvent induire en erreur notre programme. Donc cette méthode dépend beaucoup de l'environnement extérieur. La solution que l'on pourrait apporter à ce problème serait de mettre des leds en dessous du drone pour avoir un éclairage continu. Elle est adapté à la caméra du dessous du drone vu qu'aucune autre forme ne s'y présente.



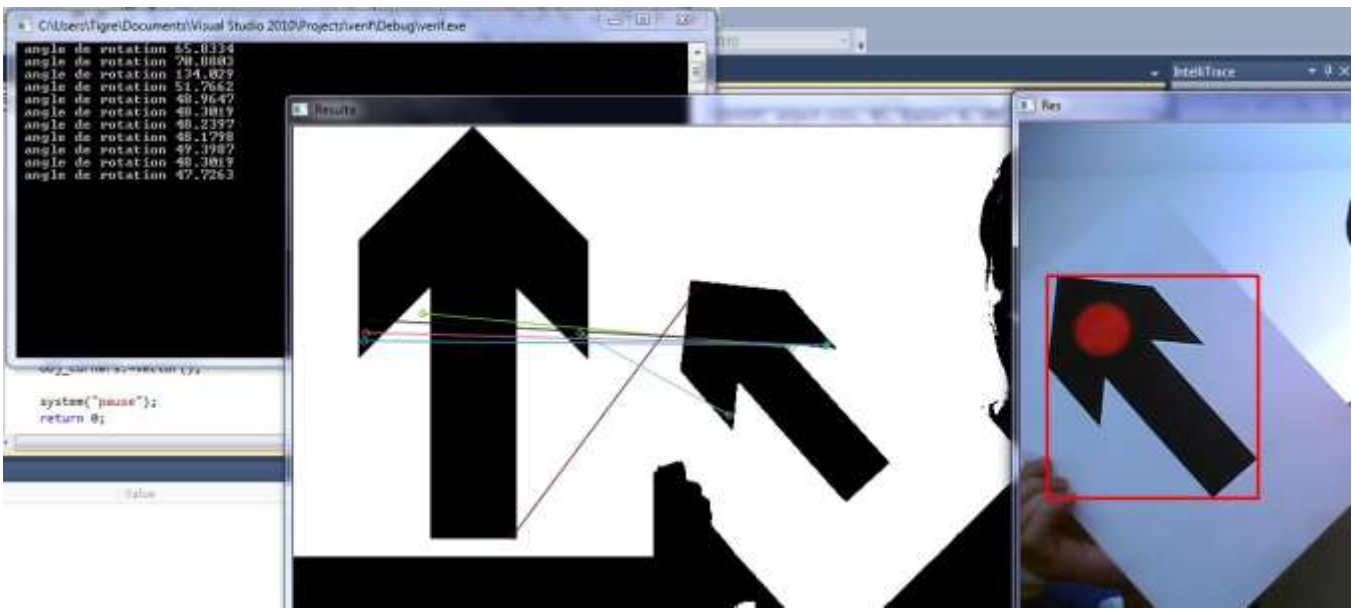
Concernant la méthode des couleurs :

Cette méthode fonctionne correctement. Le programme identifie bien le rouge, le vert et le bleu. Il détecte grâce à cela le contour de la flèche et le centre des deux cercles qui s'y trouvent. La détection de couleur se fait rapidement. Elle est donc adaptée au temps réel. Par contre, elle dépend de la luminosité (comme la méthode de détection de forme) et de la couleur de la flèche. Selon les couleurs qui se trouvent dans la flèche, il est difficile de trouver rapidement les bons filtres. La solution que l'on pourrait adopter serait la même que la détection de forme : ajouter des leds en dessous du drone et sélectionner la couleur directement sur l'image.



Concernant la méthode des points d'intérêts :

Le programme C++ fonctionne correctement. Le programme identifie bien la flèche. Les avantages de cette méthode seraient sa rapidité et sa robustesse. Ses inconvénients seraient son temps de calcul, la dépendance à une bonne luminosité et les erreurs de correspondance qui interviennent régulièrement.



Synthèse :

*Rapidité : du plus rapide au plus lent*
1. couleur
2. forme
3. point d'intérêt
4. corrélation

*Robustesse : du plus robuste au moins robuste*
1. corrélation
2. couleur
3. point d'intérêt
4. forme

# VII. Annexes

## Annexe 1 : Code de détection par corrélation

```
#include "Fonction.h"

//Fonction principale
int main()
{
	//Initilisation d'une touche de clavier
	int key = 1;
	//Flux de la webcam rangée dans capture
    //CvCapture* capture = cvCreateCameraCapture( CV_CAP_ANY );

	//Définition de 3 images
    //IplImage *src=cvCreateImage(cvSize(640,480), 8, 3);           //image saisi par
la camera

	//Chargement des 4 images
	IplImage *haut   = cvLoadImage("fleche_devant.png",1);
	IplImage *droite = cvLoadImage("fleche_droite.png",1);
	IplImage *gauche = cvLoadImage("fleche_gauche.png",1);

	//-------------------------------------SUR ECHANTILLONNAGE--------------------
------------------//

	//==================1ère méthode==============//
	//Varible qui va contenir la pourcentage du redimensionnement
	//de la nouvelle image en fonciton de l'altitude.
	int percent = 0;

	//Appelle de la fonction altitude
	percent = altitude();

	//Creation d'une nouvelle image qui sera redimensionnée en fonction de
l'altitude du drone
	//Cette fonction prends comme argument :
	// - une nouvelle taille (largeur et hauteur)
	// - une profondeur
	// - un canal
	//On garde la profondeur et le canal de l'image originale, mais on change
(cvSize) la taille de l'image
	IplImage *templ = cvCreateImage(
                                        cvSize((int)((haut-
>width*percent)/100),(int)((haut->height*percent)/100)),
                                        haut->depth,
                                        haut->nChannels );

	IplImage *templ_2 = cvCreateImage(
                                        cvSize((int)((droite-
>width*percent)/100),(int)((droite->height*percent)/100)),
                                        droite->depth,
                                        droite->nChannels );
	IplImage *templ_3 = cvCreateImage(
                                        cvSize((int)((gauche-
>width*percent)/100),(int)((gauche->height*percent)/100)),
                                        gauche->depth,
                                        gauche->nChannels );

	//On utilise ensuite cvResize pour redéfinir notre nouvelle iamge
```

```
       //Argument :
       // - image source
       // - image template
       cvResize(haut, templ);
       cvResize(droite, templ_2);
       cvResize(gauche, templ_3);

       //---------------------------------CORRELATION---------------------------------
--------//

       //Boucle infinie
    while(1)
       {

              double début;
              début = (double)getTickCount();

              //Enregistre les images du flux video dans src
              //src = cvRetrieveFrame( capture );

              FILE *file = NULL;

              file = fopen("C:/ProjetInformatique/Projet en
cours/Autre/Image/images/work.png","r");

              if(file != NULL)
              {
              IplImage *src = cvLoadImage("C:/ProjetInformatique/Projet en
cours/Autre/Image/images/work.png",1);

        //applique le filtre médian pour réduire le bruit
        cvSmooth(src,src,CV_MEDIAN,3);


//---------------------------CORRELATION-------------------------------------//

              //Creation de 4 images qui contiendra l'image resultat de chaque
correlation
              IplImage ftmp   = CreationImageFTMP(src,templ);
              IplImage ftmp_2 = CreationImageFTMP(src,templ_2);
              IplImage ftmp_3 = CreationImageFTMP(src,templ_3);

              //Creation du cradre qui va définir la zone de détection
              CvPoint cadre_pt1 = cadre_pt0(src,templ);
              CvPoint cadre_pt2 = cadre_ptbis(src,templ,cadre_pt1);

              CvPoint cadre_pt1_2 = cadre_pt0(src,templ_2);
              CvPoint cadre_pt2_2 = cadre_ptbis(src,templ_2,cadre_pt1_2);

              CvPoint cadre_pt1_3 = cadre_pt0(src,templ_3);
              CvPoint cadre_pt2_3 = cadre_ptbis(src,templ_3,cadre_pt1_3);

              //Correlation entre l'image source et l'image de référence. Rangement du
résultat dans ftmp
        cvMatchTemplate(src, templ, &ftmp, CV_TM_CCOEFF_NORMED);

              cvMatchTemplate(src, templ_2, &ftmp_2, CV_TM_CCOEFF_NORMED);

              cvMatchTemplate(src, templ_3, &ftmp_3, CV_TM_CCOEFF_NORMED);

        //retrouver dans 'ftmp' les coordonnées du point ayant une valeur maximale
        double min_val , max_val;
```

```
            double min_val_2,max_val_2;
            double min_val_3,max_val_3;

        CvPoint min_loc, max_loc;
            CvPoint min_loc_2, max_loc_2;
            CvPoint min_loc_3, max_loc_3;

            //Fonctions qui permettent de calculer le taux maximale de correlation
après le passage de la fonction cvMatchTemplate
        cvMinMaxLoc(&ftmp, &min_val, &max_val, &min_loc, &max_loc);

            cvMinMaxLoc(&ftmp_2, &min_val_2, &max_val_2, &min_loc_2, &max_loc_2);

            cvMinMaxLoc(&ftmp_3, &min_val_3, &max_val_3, &min_loc_3, &max_loc_3);

        //défnir un deuxième point à partir du premier point et de la taille de 'ftmp'
            CvPoint max_loc2 = cvPoint(max_loc.x + templ->width, max_loc.y + templ-
>height);//définir le deuxième point en fonction

                                                                    //de la taille du
template
            CvPoint max_loc2_2 = cvPoint(max_loc_2.x + templ_2->width, max_loc_2.y +
templ_2->height);

            CvPoint max_loc2_3 = cvPoint(max_loc_3.x + templ_3->width, max_loc_3.y +
templ_3->height);

            //Creation des variables qui contiendra le taux de correlation de chaque
image
            int max_val_100 = 0, max_val_100_2 = 0, max_val_100_3 = 0;

            //max_val compris entre [0;1] donc en %, il faut *100
            max_val_100   = (int)(max_val   * 100);
            max_val_100_2 = (int)(max_val_2 * 100);
            max_val_100_3 = (int)(max_val_3 * 100);

            //Creation du tableau qui va contenir les 4 valeurs de correlation
            int tab[4]={max_val_100,max_val_100_2,max_val_100_3};

            //Initialisation de 3 variables
            int max=0; //Va contenir le plus grand élément du tableau

            //Boucle qui permet de connaitre la plus grande valeur dans le tableau
                  for (int i=0 ; i < 3; i++)
                  {
                        if (tab[i]>max)
                        {
                              max=tab[i];
                        }
                  }

//----------------------Affichage de la direction de la fleche et du taux de
correlation------------------//

ofstream f("C:/ProjetInformatique/Projet en cours/Autre/Image/drone/direction.txt");

            if((max == max_val_100) && (max_val_100 > SEUIL))
            {

                  int a = 1;
                  f << a;
                  f.close();
```

```
                cout << "=== FLECHE AVANCER \x85 "<< max_val_100 <<"===" << endl;
                cout << "etat : " << a << endl;
                cvNamedWindow( "FLECHE", CV_WINDOW_AUTOSIZE );
                cvShowImage( "FLECHE", haut );
        }

        else if((max == max_val_100_2) && (max_val_100_2 > SEUIL))
        {
                int d = 2;
                f << d;
                f.close();

                cout << "=== FLECHE DROITE \x85 " << max_val_100_2<< "===" <<
endl;
                cout << "etat : " << d << endl;
                cvNamedWindow( "FLECHE", CV_WINDOW_AUTOSIZE );
                cvShowImage( "FLECHE", droite );
        }

        else if((max == max_val_100_3) && (max_val_100_3 > SEUIL))
        {
                int g = 3;
                f << g;
                f.close();

                cout << "=== FLECHE GAUCHE \x85 " << max_val_100_3<< "===" <<
endl;
                cout << "etat : " << g << endl;
                cvNamedWindow( "FLECHE", CV_WINDOW_AUTOSIZE );
                cvShowImage( "FLECHE", gauche);
        }

        else
        {
                cout << "\t=== AUCUN ===" << endl;
                cvDestroyWindow("FLECHE");
        }


    //si la valeur maximale de 'ftmp' est supérieure au 'seuil'
    //dessiner un rectangle rouge utilisant les coordonnées des deux points
'max_loc' et 'max_loc2'
    if( max_val_100 > SEUIL && max_val!=1 )
        {
                cvRectangle(src, max_loc,max_loc2, cvScalar(0,0,255));
        }

        else if( max_val_100_2 > SEUIL && max_val!=1 )
        {
                cvRectangle(src, max_loc_2,max_loc2_2, cvScalar(0,0,255));
        }

        else if( max_val_100_3 > SEUIL && max_val!=1 )
        {
                cvRectangle(src, max_loc_3,max_loc2_3, cvScalar(0,0,255));
        }


        fonctionOuvertureFenetre(src,templ_2);

        //On attends
```

```
                key= cvWaitKey(10);

                début = ((double)getTickCount() - début)/getTickFrequency();
                cout << "TEMPS : " << début << endl << endl;
                }

                Sleep(1000);
                remove("C:/ProjetInformatique/Projet en
cours/Autre/Image/images/work.png");
    }
   //DESTRUCTEUR(capture);
   return 0;
}
```

## Annexe 2 : 1ère méthode de sous-échantillonnage

```
//Variable qui va contenir la pourcentage du redimensionnement
//de la nouvelle image en fonciton de l'altitude.
int percent = 0;

//Appelle de la fonction altitude
percent = altitude();  => CF. ANNEXE 2

//Creation d'une nouvelle image qui sera redimensionnée en fonction de l'altitude
//du drone
//Cette fonction prends comme argument :
// - une nouvelle taille (largeur et hauteur)
// - une profondeur
// - un canal
//On garde la profondeur et le canal de l'image originale, mais on change (cvSize)
//la taille de l'image
IplImage *templ = cvCreateImage(
          cvSize((int)((haut->width*percent)/100),(int)((haut->height*percent)/100)),
          haut->depth,
          haut->nChannels );

IplImage *templ_2 = cvCreateImage(
          cvSize((int)((droite->width*percent)/100),(int)((droite>height*percent)/100)),
          droite->depth,
          droite->nChannels );

IplImage *templ_3 = cvCreateImage(
          cvSize((int)((gauche->width*percent)/100),(int)((gauche->height*percent)/100)),
          gauche->depth,
          gauche->nChannels );

IplImage *templ_4 = cvCreateImage(
          cvSize((int)((bas->width*percent)/100),(int)((bas->height*percent)/100)),
          bas->depth,
          bas->nChannels );

//On utilise ensuite cvResize pour redéfinir notre nouvelle image
//Argument :
// - image source
// - image template
cvResize(haut, templ);
cvResize(droite, templ_2);
cvResize(gauche, templ_3);
cvResize(bas, templ_4);
```

## Annexe 3 : fonction altitude

```
//Fonction altitude
int altitude()
{
	//Initialisation de la variable valeur
	int valeur = 0;

	//Lecture du fichier ALTITUDE.txt
	ifstream fichier("ALTITUDE.txt", ios::in);

	//Si le fichier existe
	if(fichier)
	{
		//Creation de la variable data qui contiendra la valeur
		//de l'altitude
		int data;

		//On rempli la variable data par le contenu du fichier
		fichier >> data;

		//Condition selon la valeur de l'altitude
		if(data<=500)
			valeur = 100;
		else if(500<data<1100)
			valeur = 50;
		else if(1100<data<1600)
			valeur = 25;
		else if(1600<data<2200)
			valeur = 13;
		else
			valeur = 6;

	}
	//On retourne la valeur pour qu'elle soit lu
	return valeur;
}
```

## Annexe 4 : 2ème méthode de sous-échantillonnage

```
//Initialisation de la variable x qui s'occupe de la division
int x = 0;

//Convertion de IpImage en Mat
Mat bas_conv(bas);
Mat haut_conv(haut);
Mat gauche_conv(gauche);
Mat droite_conv(droite);

//Les matrices temporaires dans lesquelles le sur-echantillonnage va être stocker
Mat bas_temp,haut_temp,gauche_temp,droite_temp;
Mat bas_temp_temp,haut_temp_temp,gauche_temp_temp,droite_temp_temp;


//Fonction de sur-echantillionnage d'apres la technique de la pyramide
//On declare la destination IplImage object avec la taille, la profondeur et la channel

pyrDown(bas_conv, bas_temp, Size( bas_conv.cols/x, bas_conv.rows/x ));//divise par 2 la taille de la fleche
pyrDown(haut_conv, haut_temp, Size( haut_conv.cols/x, haut_conv.rows/x ));
pyrDown(gauche_conv, gauche_temp, Size( gauche_conv.cols/x, gauche_conv.rows/x ));
pyrDown(droite_conv, droite_temp, Size( droite_conv.cols/x, droite_conv.rows/x ));
```

```
//On convertie l'image MAt en IplImage grâce à cvCloneImage en castant
IplImage *templ   = cvCloneImage( &(IplImage)haut_temp );
IplImage *templ_2 = cvCloneImage( &(IplImage)droite_temp );
IplImage *templ_3 = cvCloneImage( &(IplImage)gauche_temp );
IplImage *templ_4 = cvCloneImage( &(IplImage)bas_temp );
```

## Annexe 5 : Programme Node JS de détection par corrélation

```
var arDrone = require('ar-drone');
var client = arDrone.createClient();

//################ PILOTAGE DU DRONE ###############//
//Vitesse du drone comprise entre 0 et 1
/*
client.up(0.5); //Monter
client.down(0.5); // Descendre
client.front(0.5); //Avancer
client.back(0.5); //Reculer
client.takeoff(); // Décoller
client.land(); // Atterrir
client.right(0.5); //Translater à droite
client.left(0.5); //Translater à gauche
client.counterClockwise(0.5); //Tournoi à droite
client.clockwise(0.5); //Tournoi à gauche
client.stop(); // S'arrêter
client.getVideoStream(); //Obtenir la vidéo du drone
client.getPngStream(); // Obtenir une image PNG du drone
*/

//################## PNG ##################//

console.log('Connection png program!');

//Libraries
var fs            = require('fs');
var pngStream = client.getPngStream();

//PNG
var tick = 0;
var sto = 0;
        var lastPng;
        pngStream
          .on('error', console.log)
          .on('data', function(pngBuffer) {
                if (!(tick++ % 5)) {
                        lastPng = pngBuffer;
                        fs.writeFileSync("./images/img" + sto++%6 + ".png", lastPng);
                        //fs.writeFileSync("./currentimg.txt", sto%6);
                        fs.exists("./images/img/work.png", function (e)
```

```
                        {
                                if (!e) {
                                        fs.writeFileSync("./images/work.png", lastPng);
                                }
                        });
                }
        });

//################### CONTROL ####################//

console.log('Connection control program!');

var io = require('socket.io');
var evt = require('events');
var ev = new evt.EventEmitter();


setInterval(function()
{
        console.log("#####################################");

        fs.readFile("./drone/direction.txt", "UTF-8",function(err, contenu)
        {
                console.log(contenu);
                if(err)
                        console.log(err);

                if(contenu == 1){
                        ev.emit("go");
                }

                else if(contenu == 2)
                {
                        ev.emit("droite");
                }

                else if(contenu == 3)
                {
                        ev.emit("gauche");
                }

                else
                        console.log("RIEN\n");
        });
},1000);

ev.on("go", function(){
                client.stop();
                console.log("Je vais tout droit");
```

```
                client.front(0.01);
});

ev.on("droite", function(){
                client.stop();
                console.log("Je tourne à droite");
                setTimeout(function(){
                client.counterClockwise(1);
                },1000);
});

ev.on("gauche", function(){
                client.stop();
                console.log("Je tourne à gauche");
                setTimeout(function(){
                client.clockwise(1);
                },1000);
});

//#################### KEY ########################//

console.log("Connection key program");

var keypress = require('keypress');
var STDIN = process.stdin;
keypress(STDIN);

var keys = {
  'space': function(){
        console.log('Takeoff!');
   client.takeoff();
 },

  'escape': function(){
        console.log('Land!');
   client.stop();
   client.land();
        },

        'up' : function(){
        console.log("UP");
        client.up(0.1);
        },

        'p' : function(){
        console.log("PAUSE");
        client.pause();
        }
 }
```

```
var quit = function(){
  console.log('Quitting');
  STDIN.pause();

  client.stop();
  client.land();
  client._udpControl.close();
 }

STDIN.on('keypress', function (ch, key) {
  if(key && keys[key.name]){ keys[key.name](); }
  if(key && key.ctrl && key.name == 'c') { quit(); }
});


STDIN.setRawMode(true);
STDIN.resume();
```