Ce dm était composé de 2 exercices, un permettant de calculer le nombre de globules dans une image et le second, où nous devions retrouver une zones sélectionnée sur une image, sur une seconde image prise à un endroit très proche de la première…

Exercice 1 :

Le but de cette exercice était de comptabiliser le nombre de globules sur l'image ci-dessous.

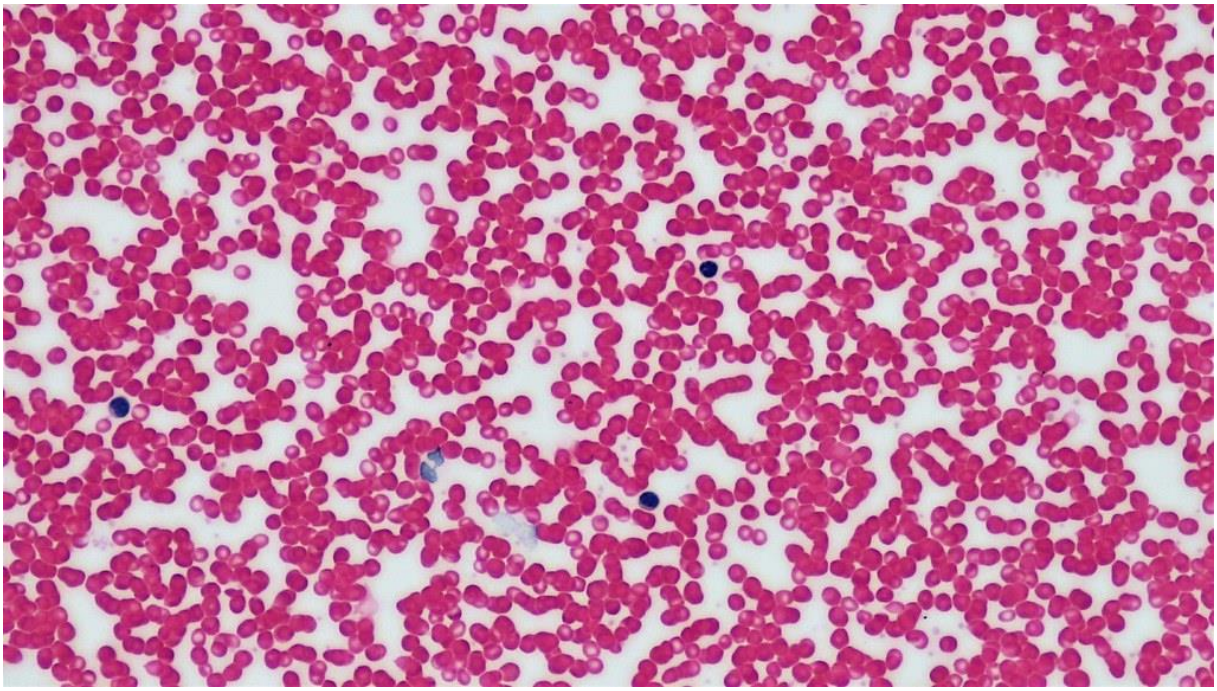

Figure 1 : image rempli de globule à compter

Pour se faire, j'ai utilisé 2 méthodes différentes : un comptage rapide et un autre plus précis.

Tout d'abord, j'ai dilater puis éroder deux fois, afin d'obtenir de meilleurs résultats. J'ai donc utilisé les fonctions imdilate() et imerode().

Pour un comptage rapide, nous pouvons utilisé la fonction suivante :

```
b = bwconncomp(erode);
nombre_de_zone1 = b.NumObjects;
```

Elle nous permet de compter le nombre de zones dans l'image, l'image doit être en noir et blanc…

Cependant ce comptage n'est pas précis, puisque certains globules sont les uns au-dessus des autres. Grâce à ce comptage, j'ai obtenu 544 cellules.

J'ai alors essayé une autre méthode : tout d'abord, j'ai parcouru l'image, et j'ai compter le nombre de pixels noirs (correspondants au globules). Puis j'ai choisi dans l'image une zone où il y avait un globules seuls et j'ai compter combien fallait il de pixels pour représenter un globule…

Ensuite, j'ai fait la division du nombre de pixel noir par le nombre de pixels nécessaires pour un globules, et j'obtiens alors 794 globules dans cette image !

```
Première méthode:
 Avec un comptage rapide, il y en a :
   544

Avec la seconde méthode (nombre de pixels noirs divisé par l aire en pixel d un globule, il y a
   794
```

Figure 2 : impression d'écran de la console Matlab.

Bien que les deux résultats soit différents, ils restent dans le même ordre de grandeur, ce qui est cohérents. Le premier comptage n'étant qu'un comptage rapide, la seconde méthode doit s'approcher du véritable nombre de globules…

Exercice 2 :

Cet exercice nous permet de mettre en œuvre des processus utilisé en vision robotique : en effet, les robots possèdent généralement deux caméras assez proche l'une de l'autre, ce qui permet d'avoir, un peut comme la vision humaine, une vision en 3D.

Pour ce faire, grâce à ginput, on laisse l'utilisateur choisir un point sur la première image, on créer alors une image réduit autour de ce point.

J'ai créer dans mon code une image réduite de 40 pixels d'intervalles autour de mon point…

Grâce au site de Matlab, on sait qu'il existe une fonction de corrélation qui va nous permettre de retrouver facilement le point sur la seconde image :

```
c = normxcorr2(imred_gray, BW2);
[y2, x2] = find(c==max(c(:)));
```

Une fois que l'on a récupéré x2 et y2, il ne nous reste plus qu'a faire une rectangle autour de ce point…

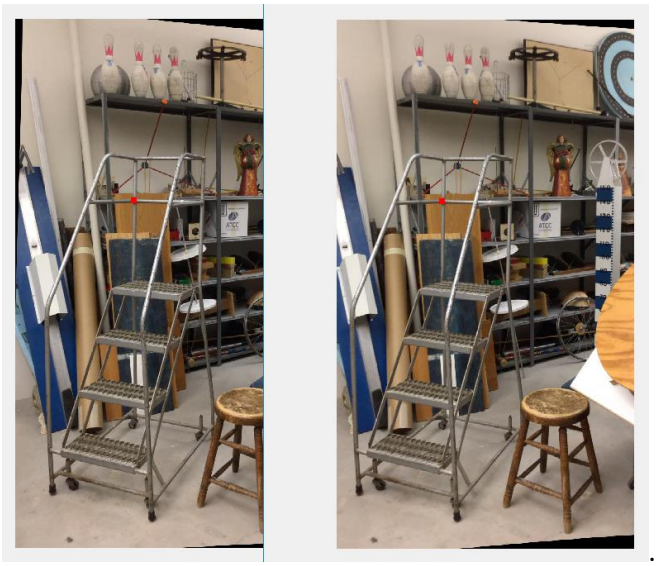

Figure 3 : point rouge sur nos deux images

Grâce à la figure 4, on voit que le programme fonctionne : on retrouve alors bien le point où l'on avait click sur la seconde image….