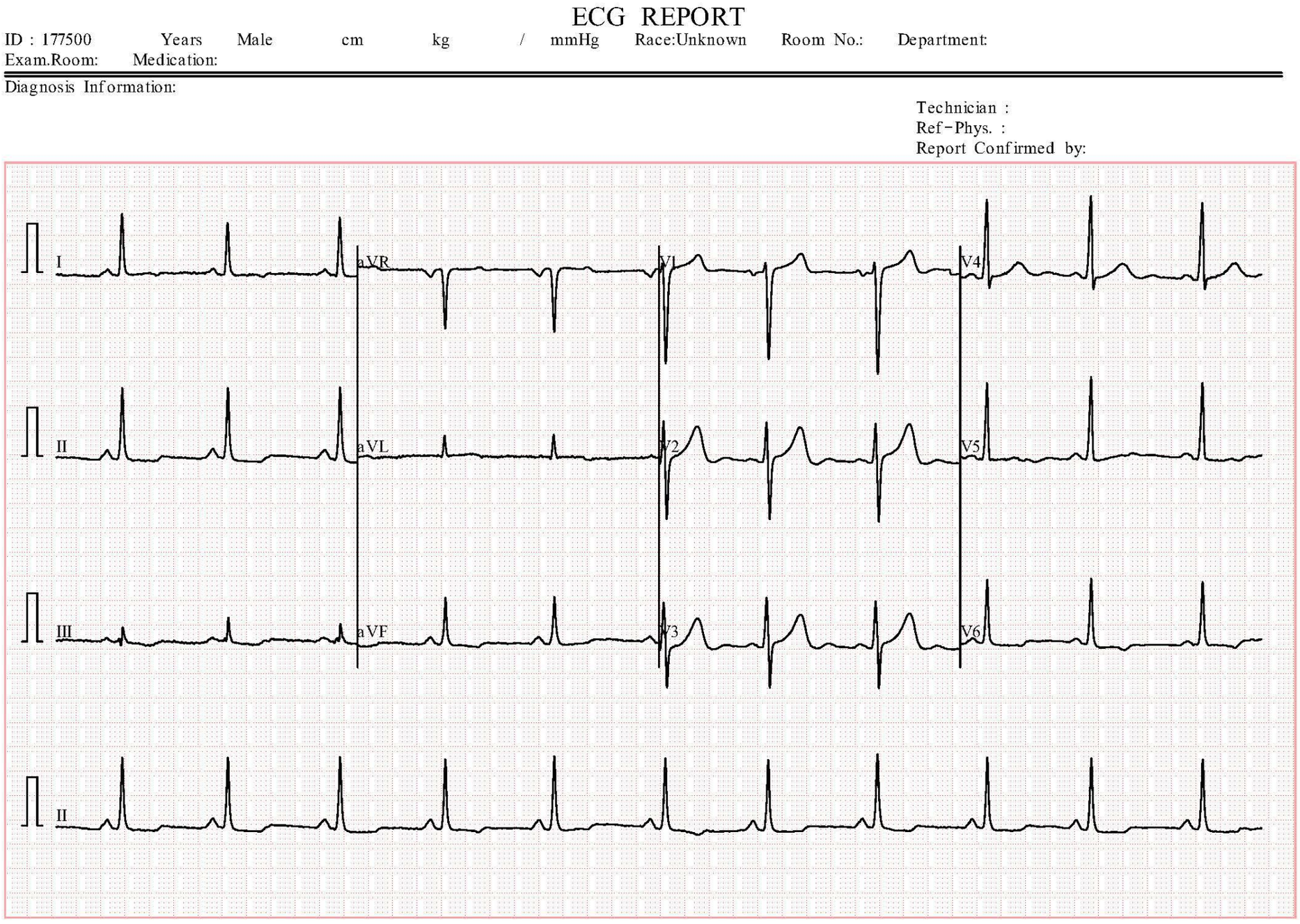
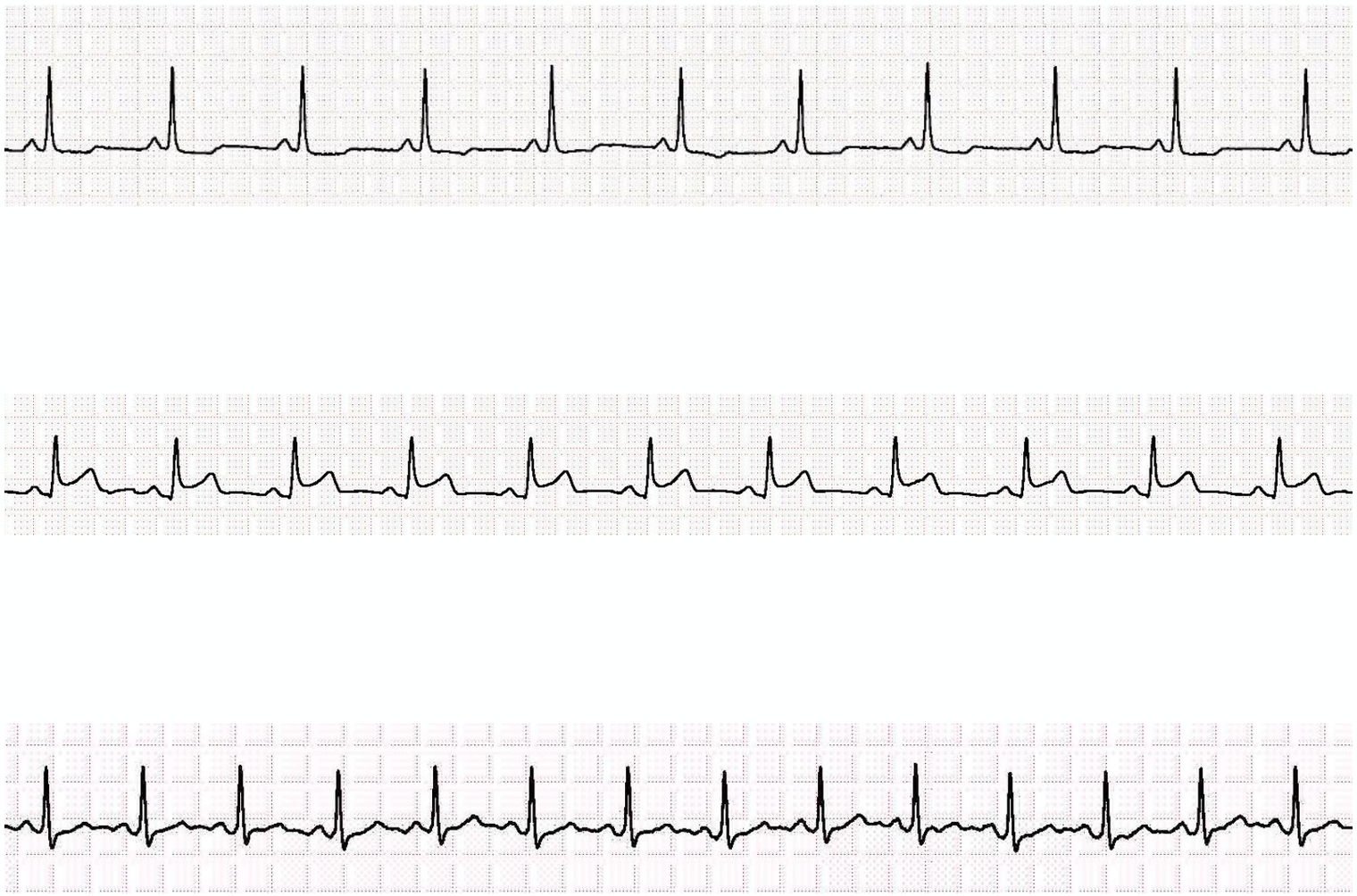
L’électrocardiogramme

Qu’est-ce qu’un électrocardiogramme ?



Objectif



Sommaire

Modélisation

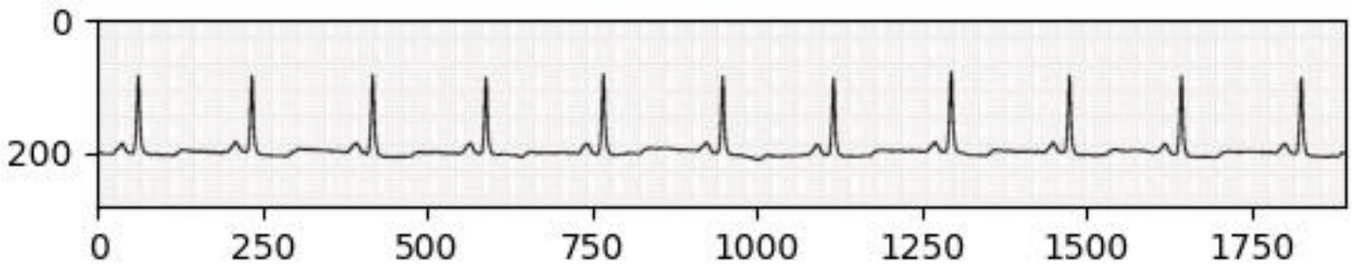
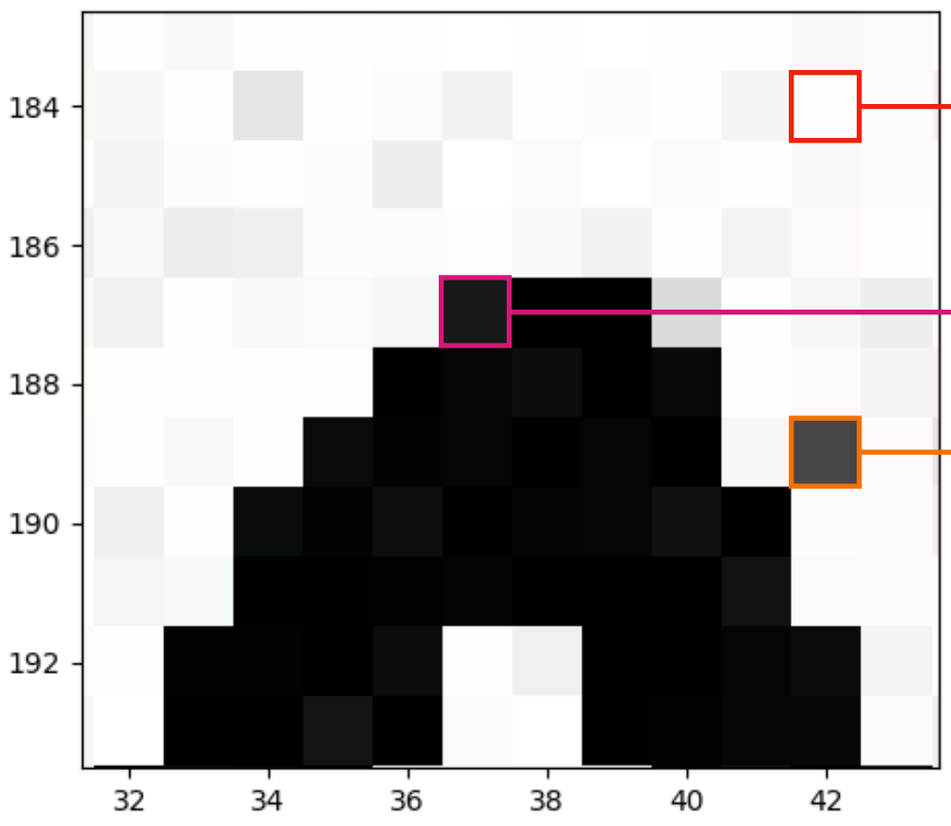


Tableau de Pixels



[R;V;B]

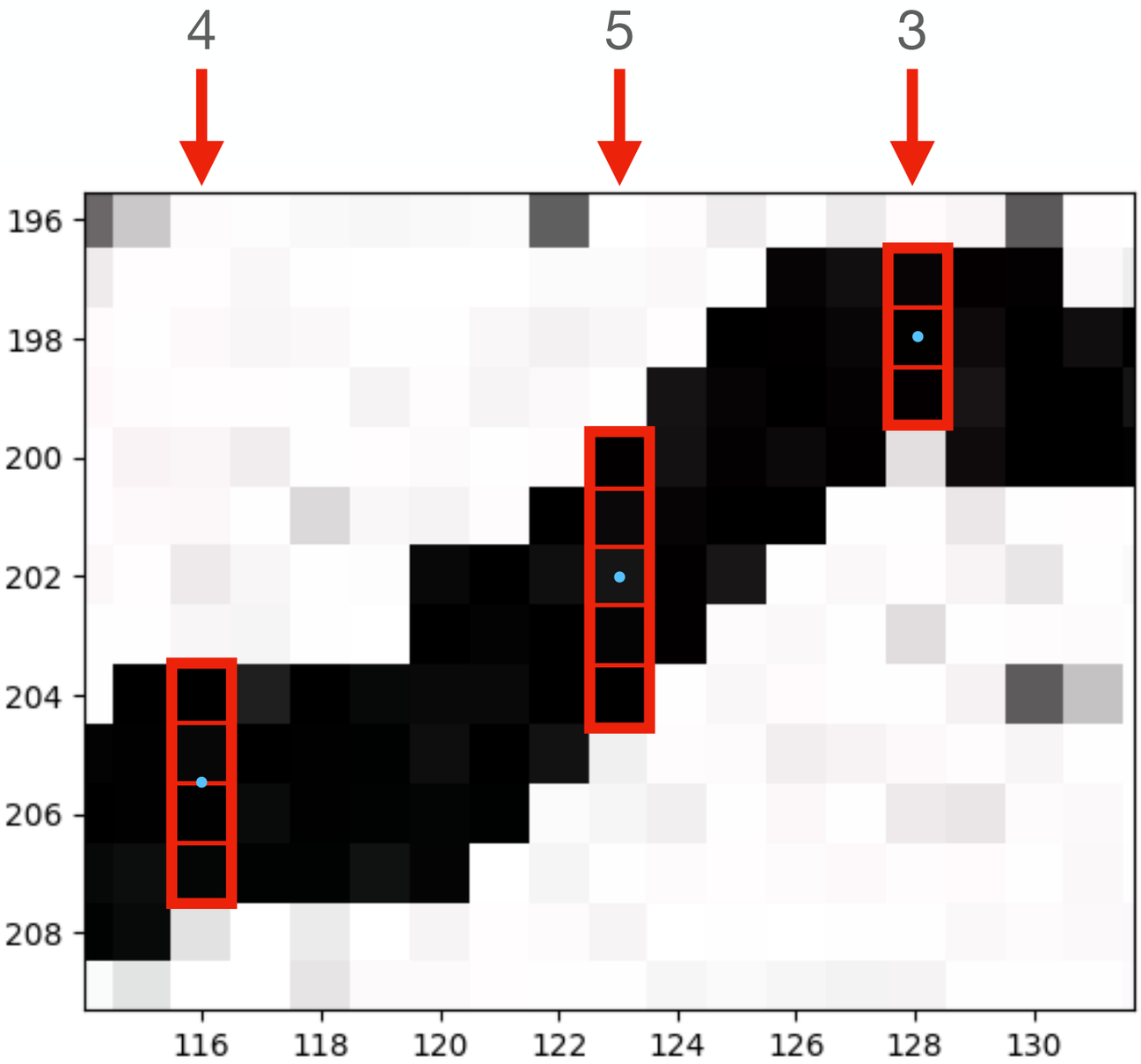
0<=R<=1
0<=V<=1
0<=B<=1

[R<0,1;V<0,1;B<0,1] : Accepté

R>0,1 ou V>0,1 ou B>0,1 : Refusé

Traitement de l’image

Récupération des Données



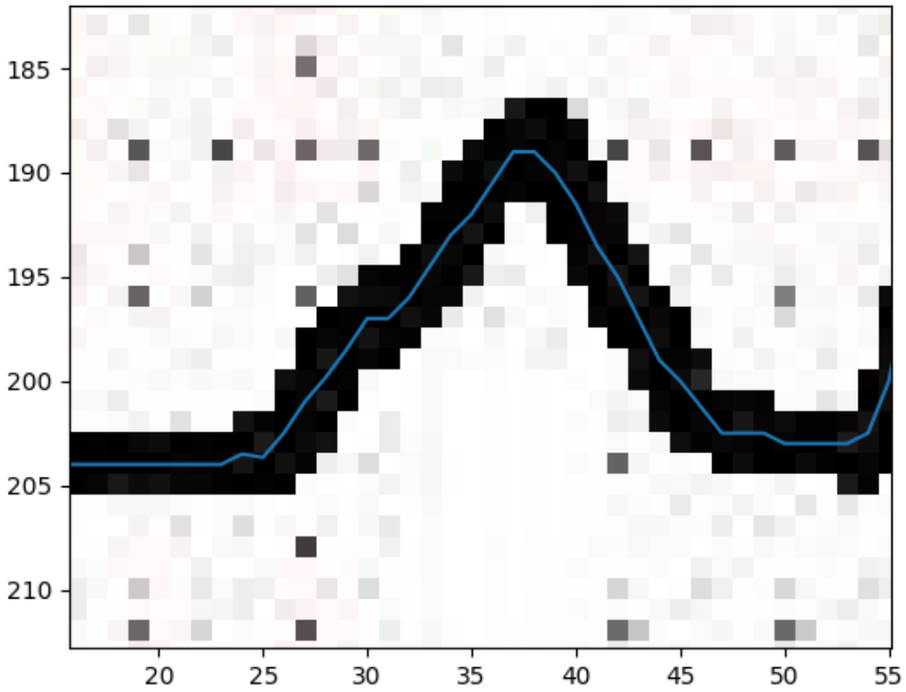
[[Abscisses],[Ordonnées]]

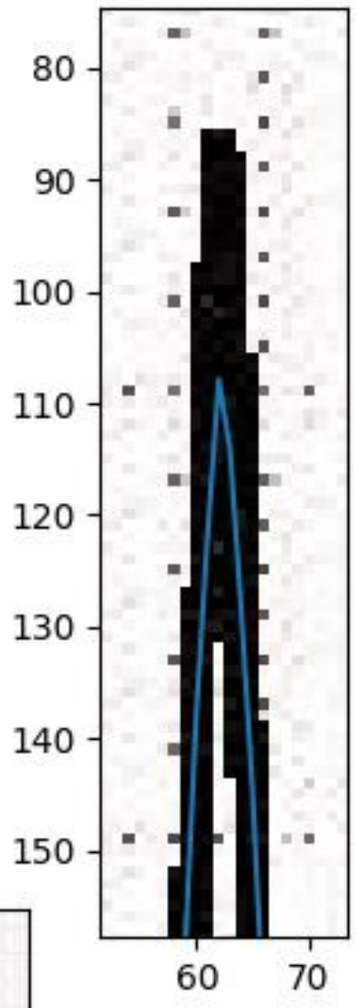
Fonction ‘electro_tableau’ (ligne 97)

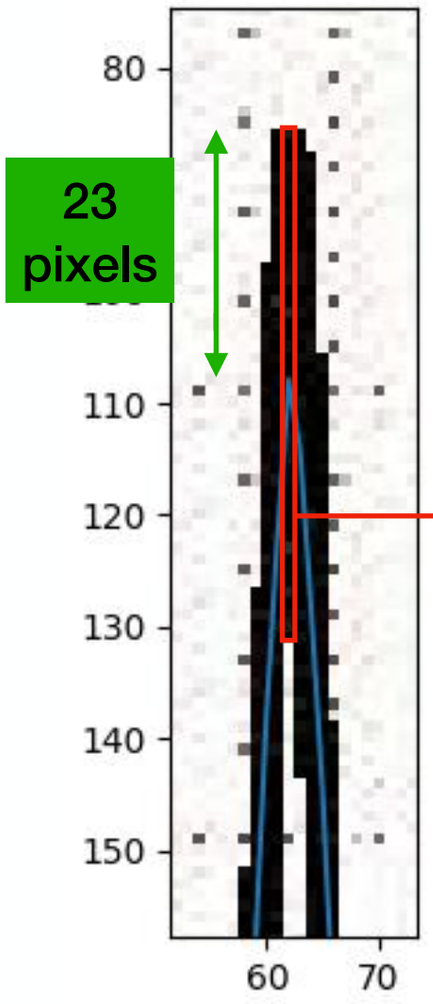
[[116., 117., 118., 119., 120., 121., 122., 123.],
[205.5, 206. , 205.5, 205.5, 204.5, 204. , 203. , 202.]]

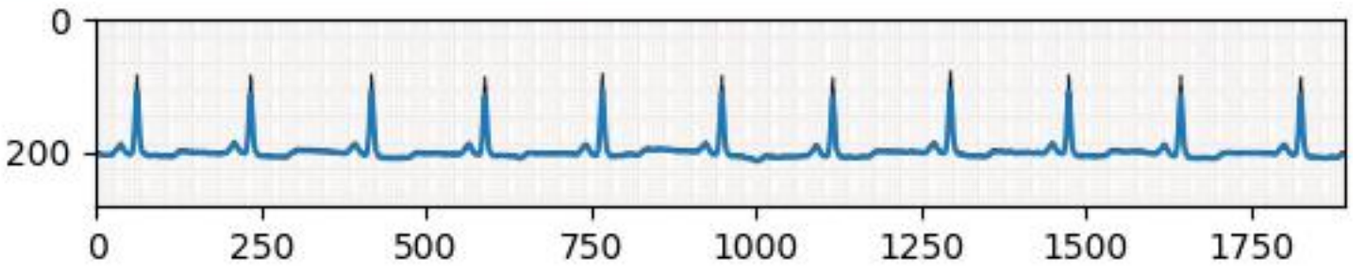
Traitement de l’image

Récupération des Données



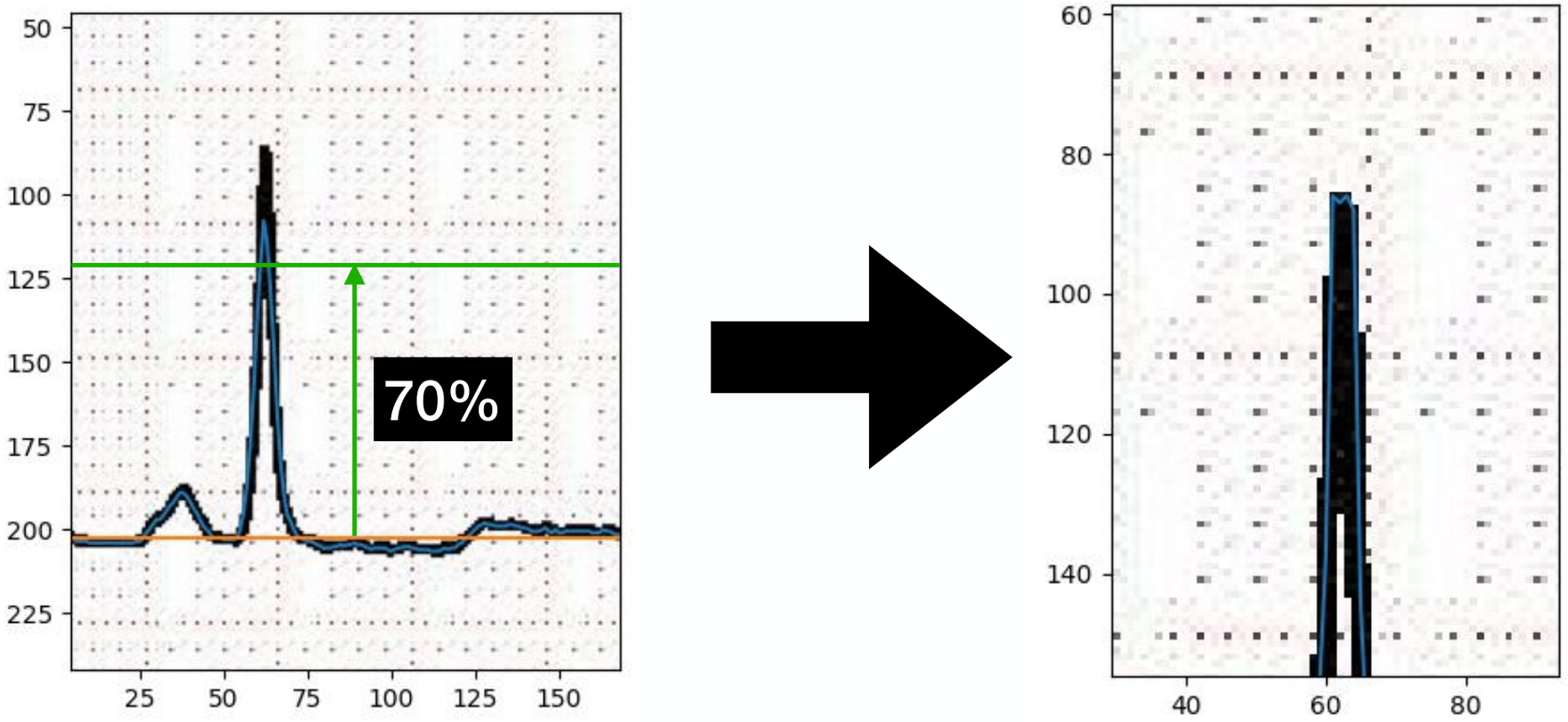




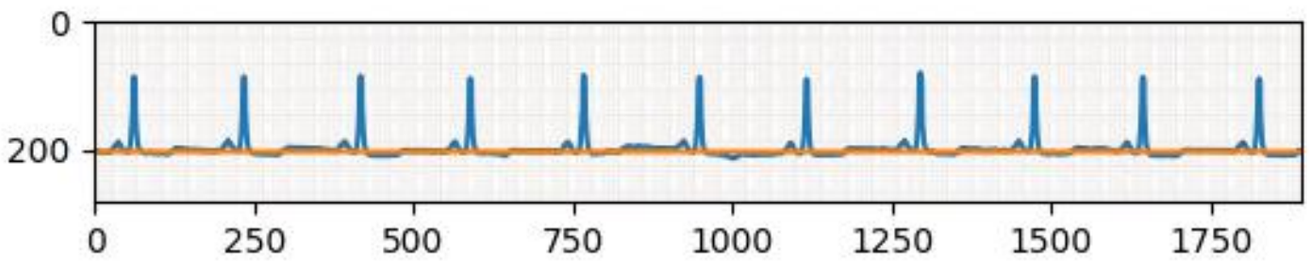


Traitement de l’image

Affinage des Données

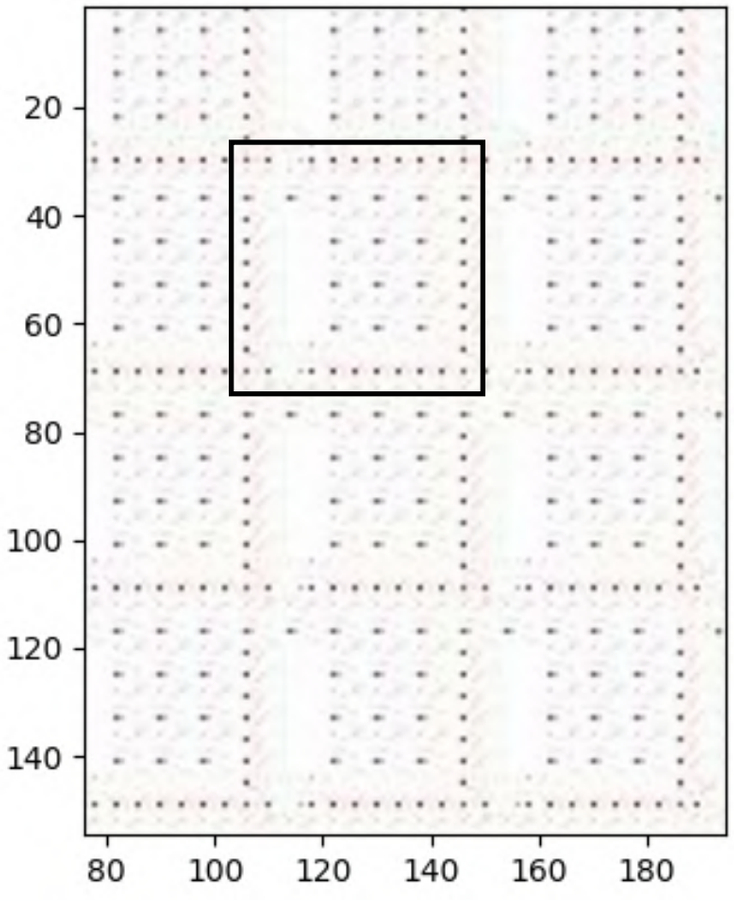


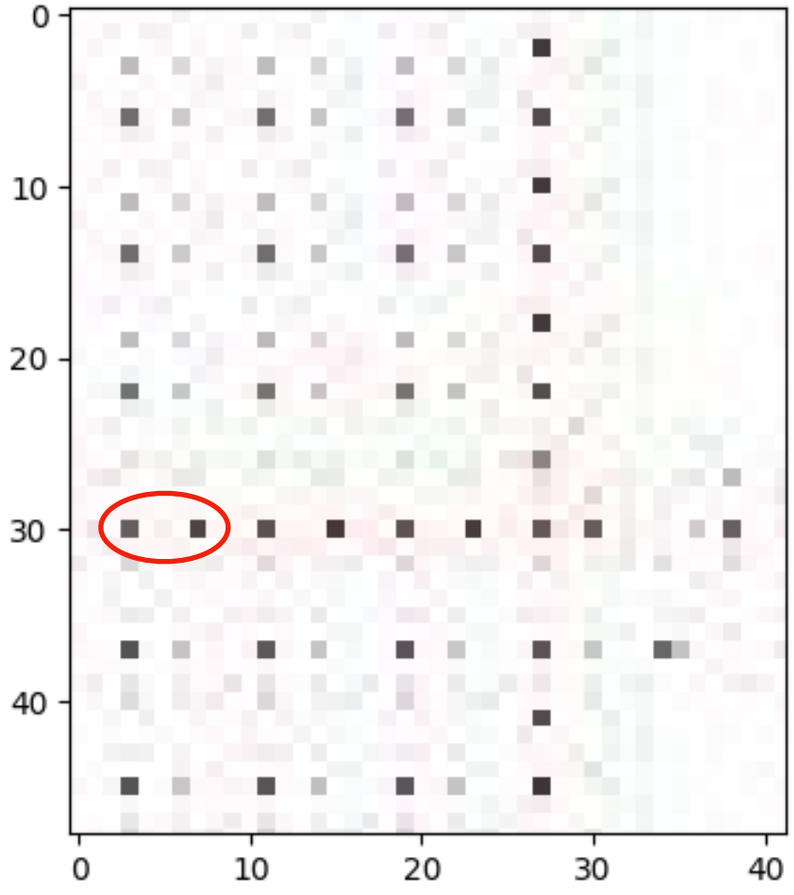
Fonction ‘affinage_tableau’ (ligne 115)

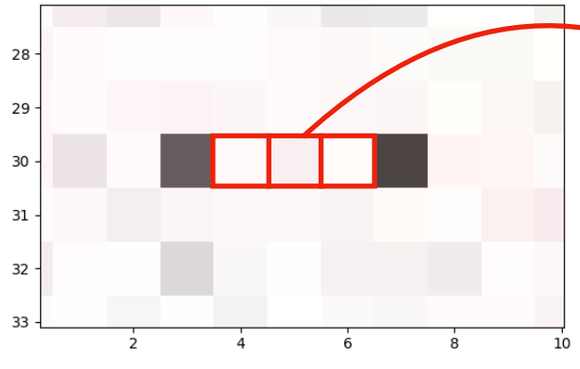


Traitement de l’image

Appliquer l’échelle de l’ECG



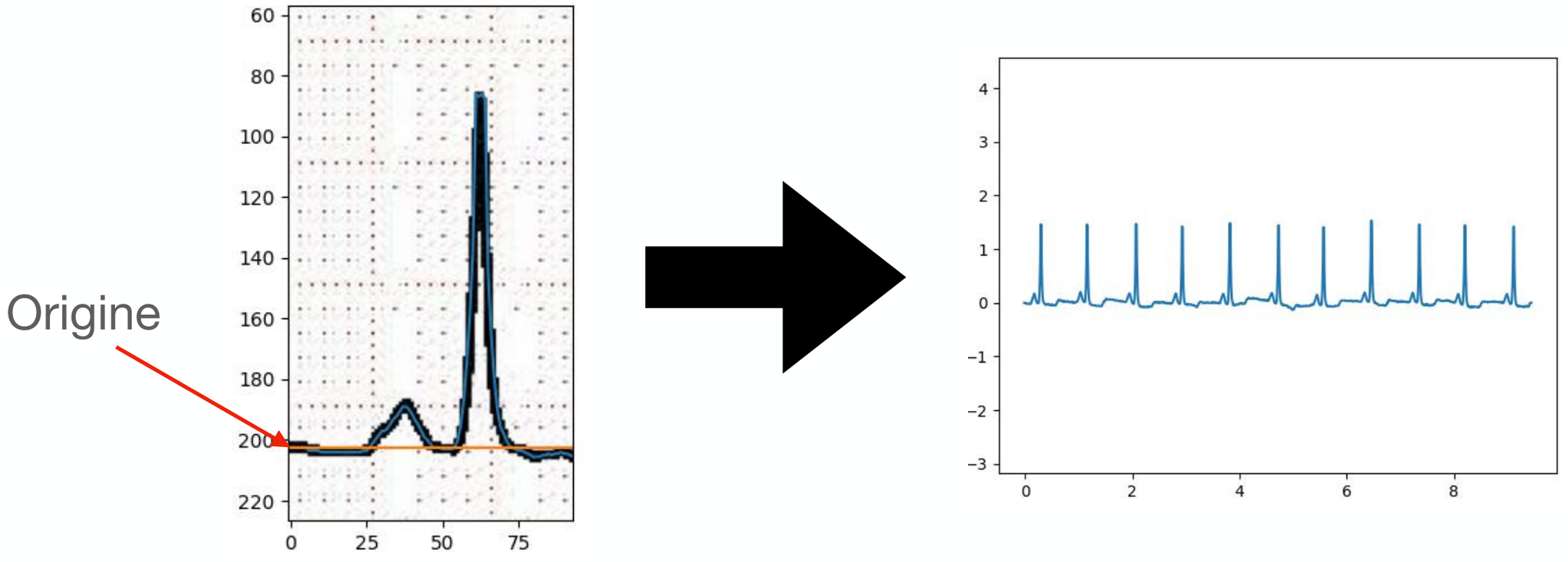




Traitement de l’image

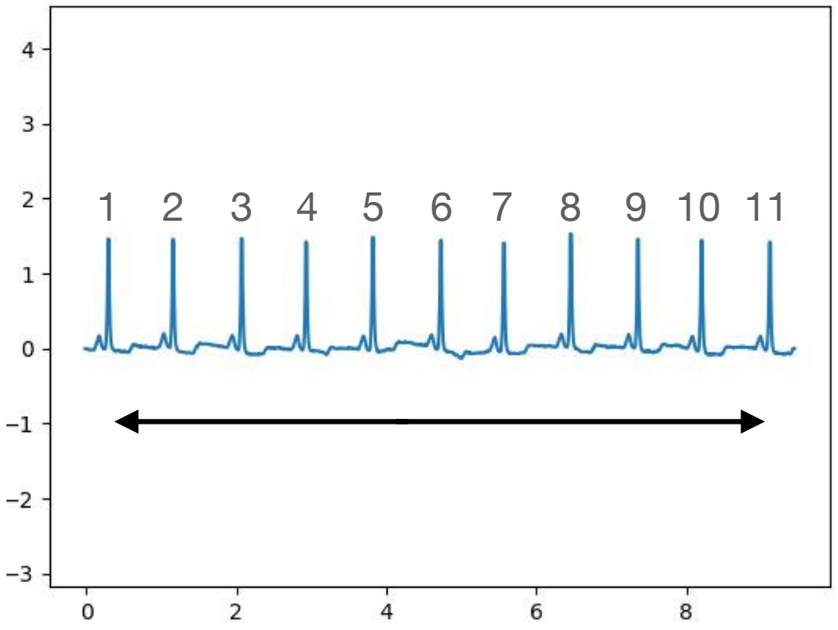
Valeurs Réelles

Fonction ‘valeur_réel_tableau’ (ligne 136)



Détermination des Informations

Calcul de la Fréquence Cardiaque



→ [Abscisse Pics]

Variable ‘L’ dans la fonction ‘fréquence’ (ligne 208)

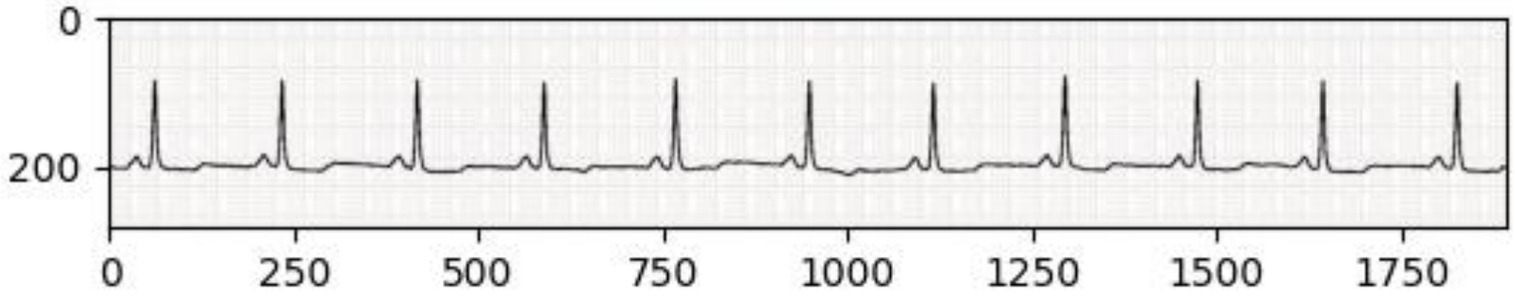
[0.29, 1.155, 2.07, 2.9250000000000003, 3.815,
4.7250000000000005, 5.560000000000005, 6.46, 7.355, 8.2, 9.11]

Fréquence Cardiaque = ((Nombre de Pics - 1) / Temps) * 60 min^{-1}

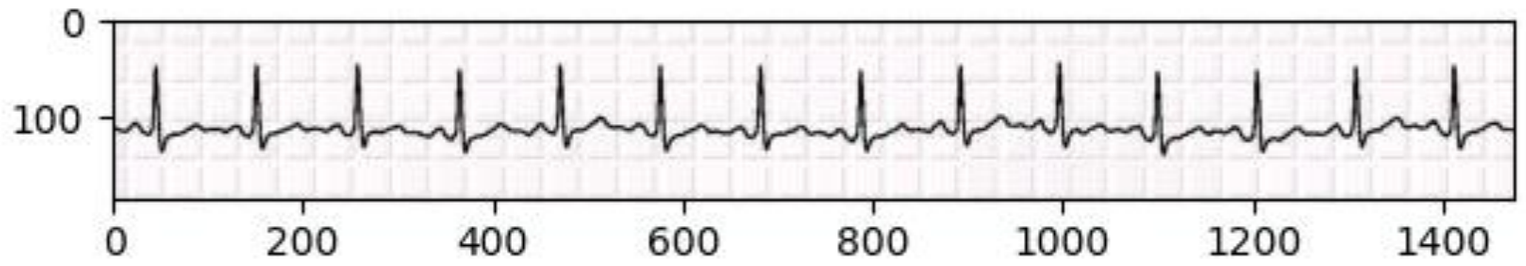
FC < 50	**50 < FC < 60**	**60 < FC < 90**	**90 < FC < 100**	**100 < FC**
Bradycardie	Suspicion de Bradycardie	Fréquence Cardiaque OK	Suspicion de Tachycardie	Tachycardie

Détermination des Informations

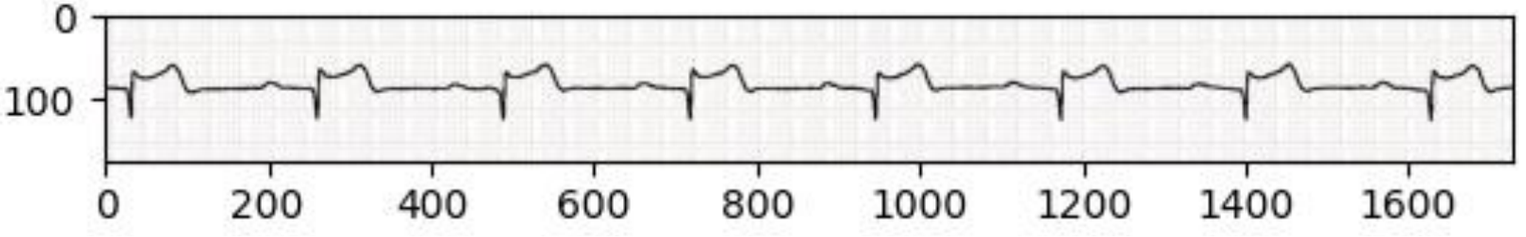
Exemples




```
68.02721088435374
Fréquence Cardiaque OK
```




```
114.20204978038068
Tachycardie - Consu
```

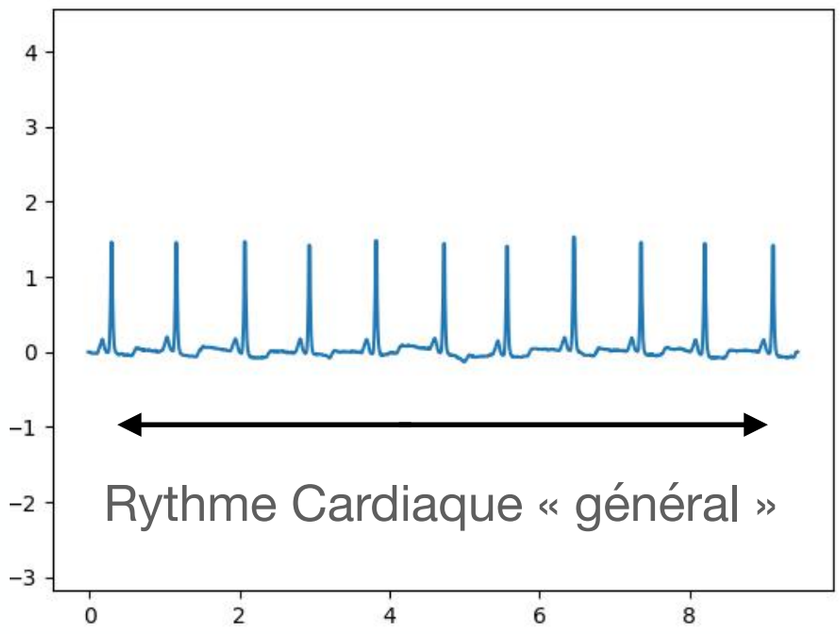



```
52.63157894736841
Suspicion de Bradycardie
```

Détermination des Informations

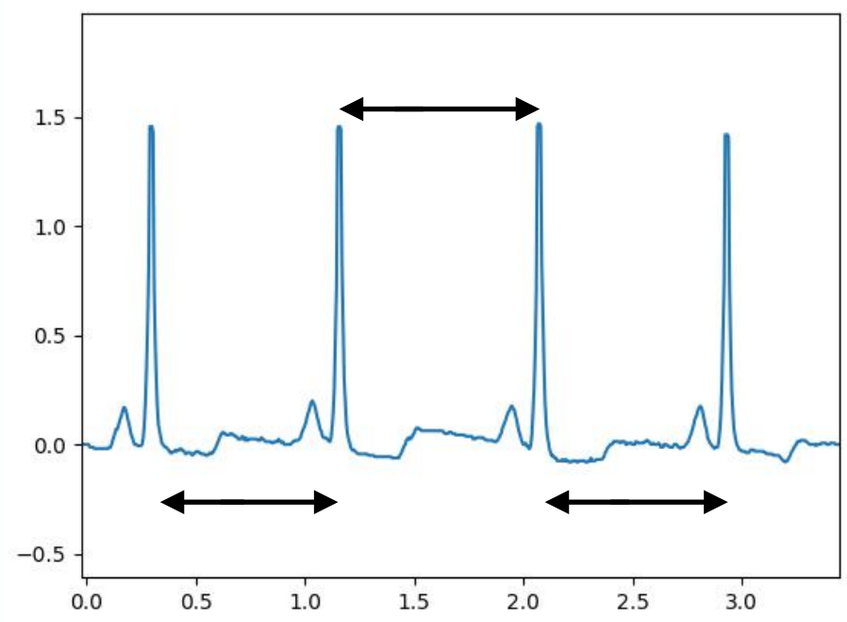
Vérification du Rythme Cardiaque

Objectif : Déterminer si les battements sont réguliers ou non



0.882 s

Fonction ‘rythme_cardiaque’ (ligne 225)



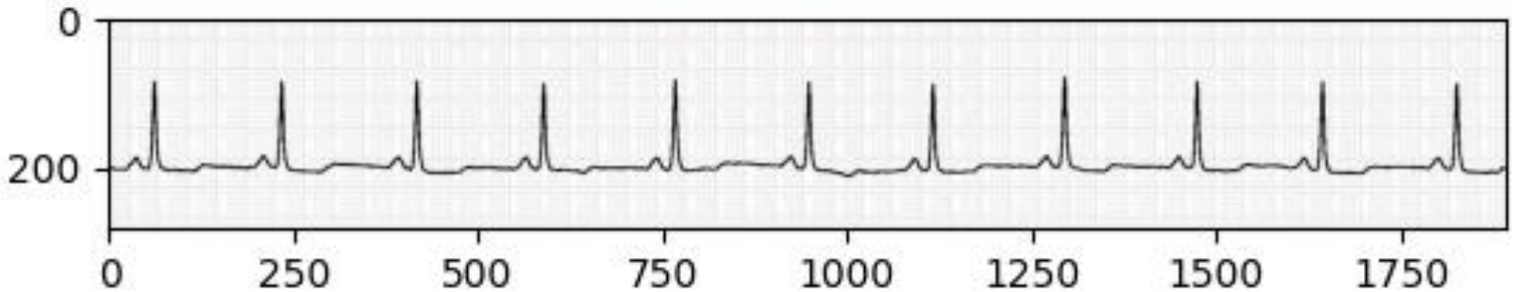
[0.8550000000000004, 0.889999999999997, 0.9100000000000006]

Variable ‘pres’ dans la fonction ‘fiabilite_RC’ (ligne 229)

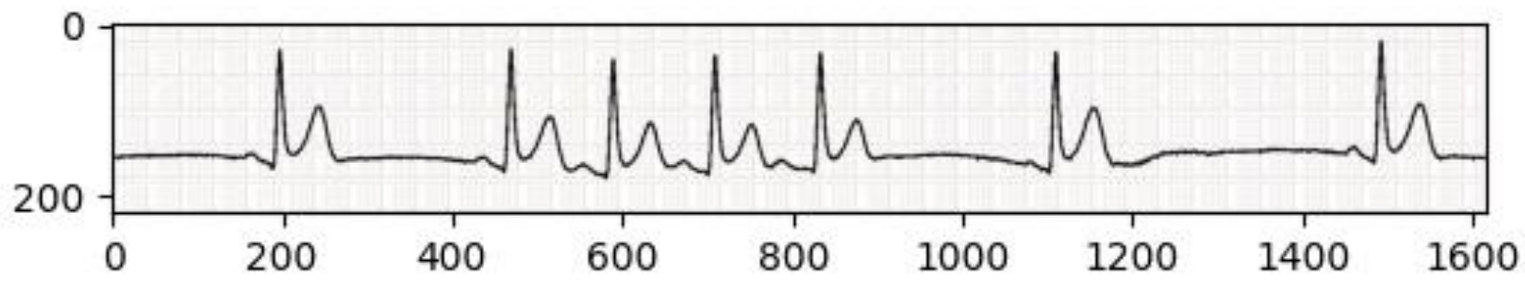
Battements Réguliers	Battements Irréguliers
Bon rythme Cardiaque	Rythme Cardiaque anormal

Détermination des Informations

Exemples



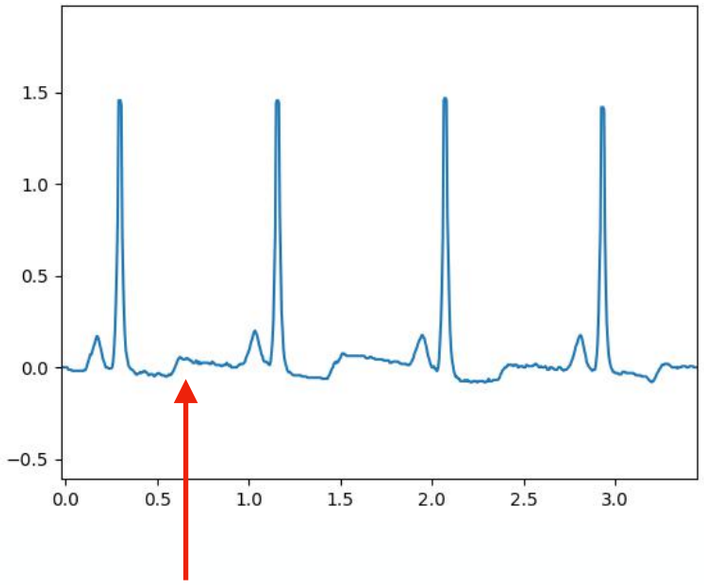

```
 0.882 Fiable
Rythme cardiaque régulier
```



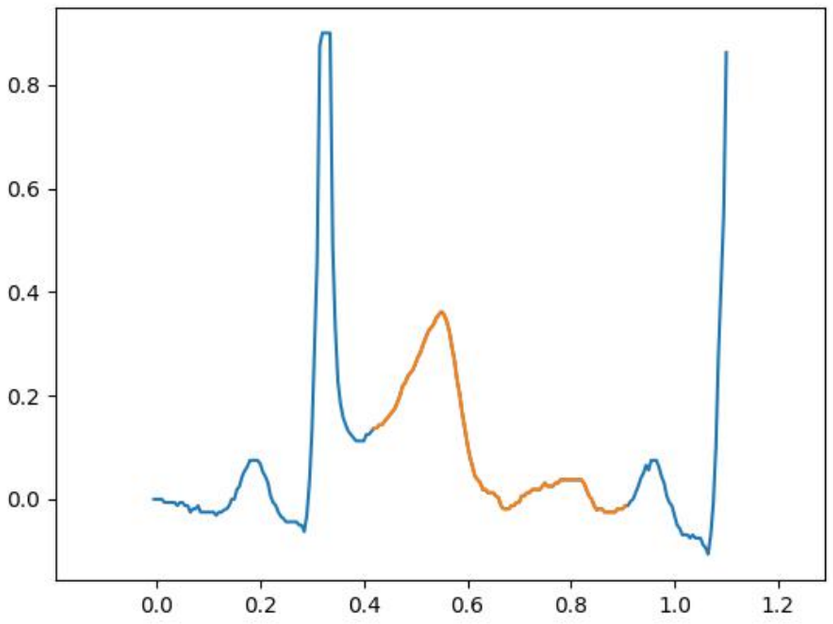

```
 1.08 Non Fiable
Rythme cardiaque anormal
```

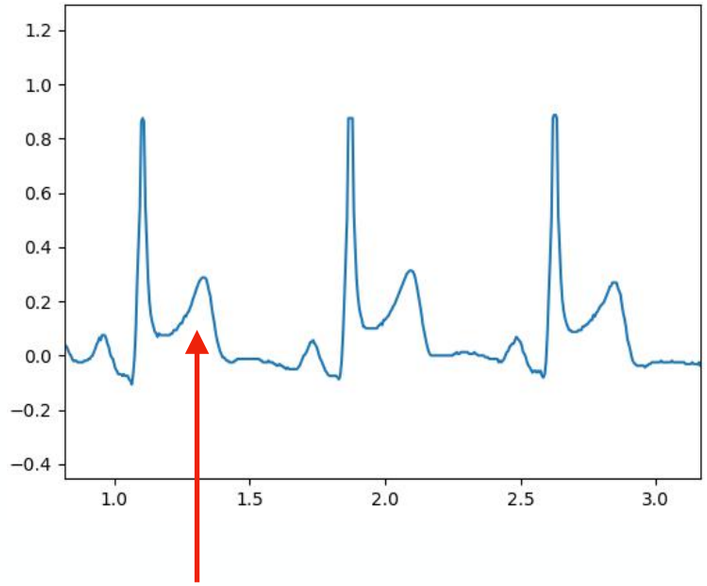
Détermination des Informations

Détection des Infarctus du Myocarde



Onde T



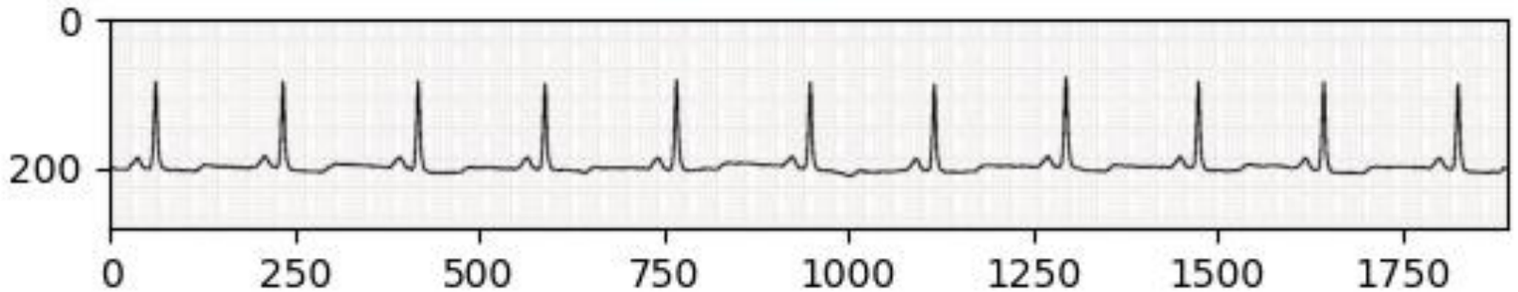


Onde T

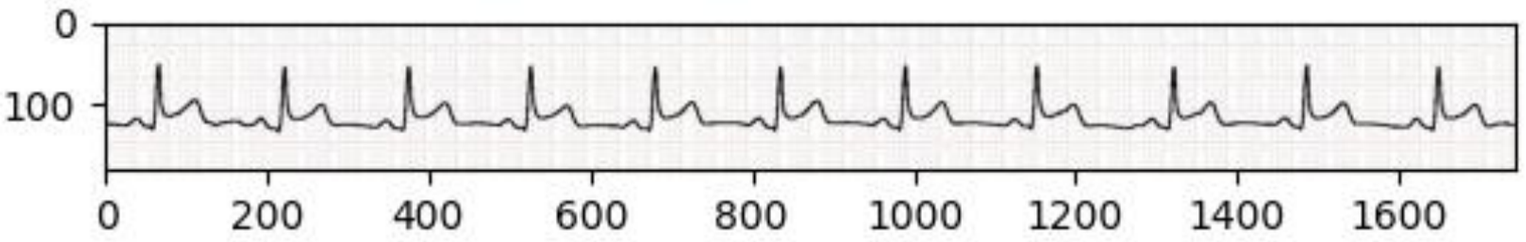
Fonction ‘anomalie_myocarde’ (ligne 240)

Détermination des Informations

Exemples



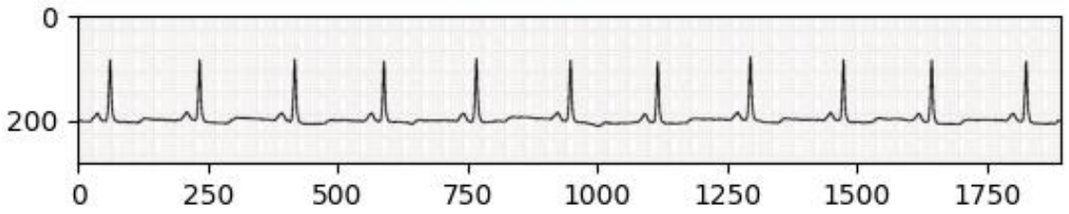

```
Pas d'Infarctus du Myocarde détecté
```




```
Infarctus du Myocarde détecté
```

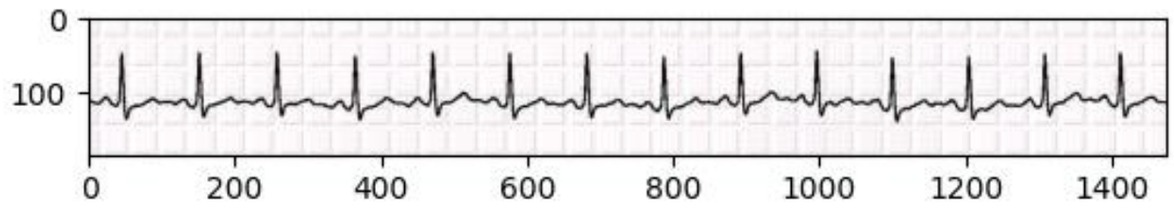
Détermination des Informations

Résultats



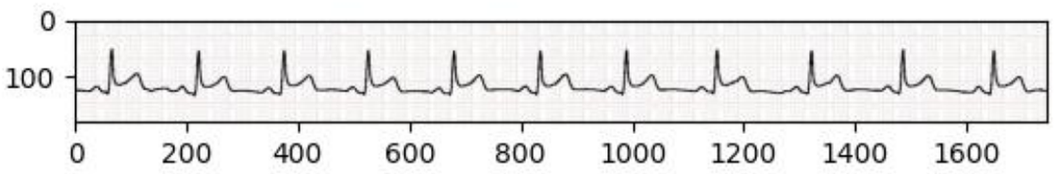

```
 68.02721088435374
Fréquence Cardiaque OK - Passez une Bonne Journée

 0.882 Fiable
Rythme cardiaque régulier
Pas d'Infarctus du Myocarde détecté
[Finished in 4.9s]
```



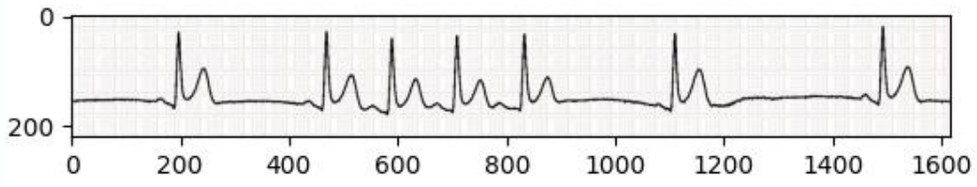

```
 114.20204978038068
Tachychardie - Consultez vite un Cardiologue

 0.5253846153846153 Fiable
Rythme cardiaque régulier
Pas d'Infarctus du Myocarde détecté
[Finished in 3.5s]
```




```
 75.80543272267846
Fréquence Cardiaque OK - Passez une Bonne Journée

 0.7915 Fiable
Rythme cardiaque régulier
Infarctus du Myocarde détecté
[Finished in 5.9s]
```




```
 55.55555555555555
Suspicion de Bradychardie - Prise de Rendez-vous chez un Cardiologue vivement conseillé

 1.08 Non Fiable
Rythme cardiaque anormal
Infarctus du Myocarde détecté
[Finished in 38.7s]
```

Analyse Spectrale avec les Séries de Fourier

Détermination des Coefficients de Fourier

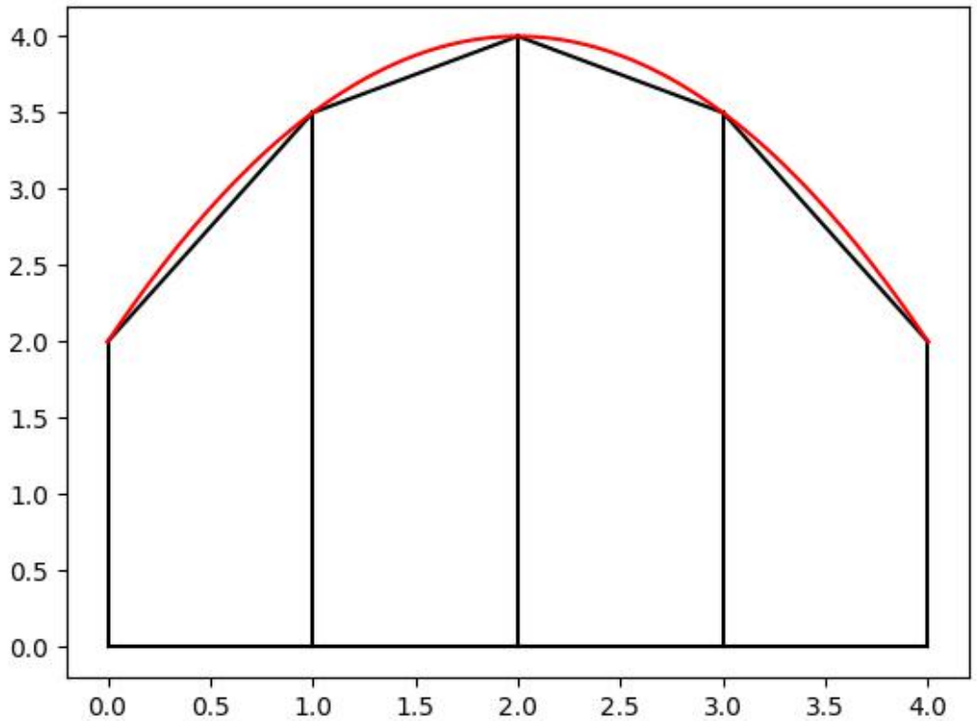
$$S_n(f) = \sum_{k=0}^{n} H_k \qquad H_n : x \mapsto a_n(f)cos(nx\frac{2\pi}{T}) + b_n(f)sin(nx\frac{2\pi}{T})$$

$$a_0(f) = \frac{1}{T}\int_0^T f(t)dt \qquad \forall n > 0, \quad a_n(f) = \frac{2}{T}\int_0^T f(t)cos(nt\frac{2\pi}{T})dt$$

$$b_0(f) = 0 \qquad \forall n > 0, \quad b_n(f) = \frac{2}{T}\int_0^T f(t)sin(nt\frac{2\pi}{T})dt$$

Analyse Spectrale avec les Séries de Fourier

Calcul des intégrales : Méthode des Trapèzes



$$Aire_{trapeze} = \frac{(g(a_k) + g(a_{k+1})) \times pas}{2}$$

$$\int_a^b g(t)dt \approx \frac{b-a}{n} \sum_{k=0}^{n-1} \frac{g(a_k) + g(a_{k+1})}{2}$$

Analyse Spectrale avec les Séries de Fourier

Coefficients de Fourier

De n=0 à n=4:

coeff_a = [0.0613733384501775, -0.02894458073373279 4, -0.0889977813545058 1, 0.0888808337254481 4, -0.0419116020494021 8]

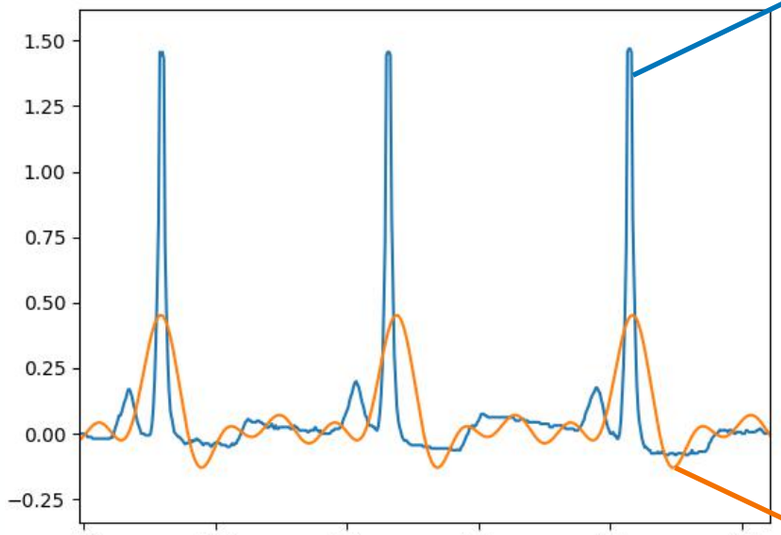
coeff_b = [0, 0.0998636679738108 5, -0.0899772853972162 8, -0.0032352163175895267, 0.0647226869164238 5]

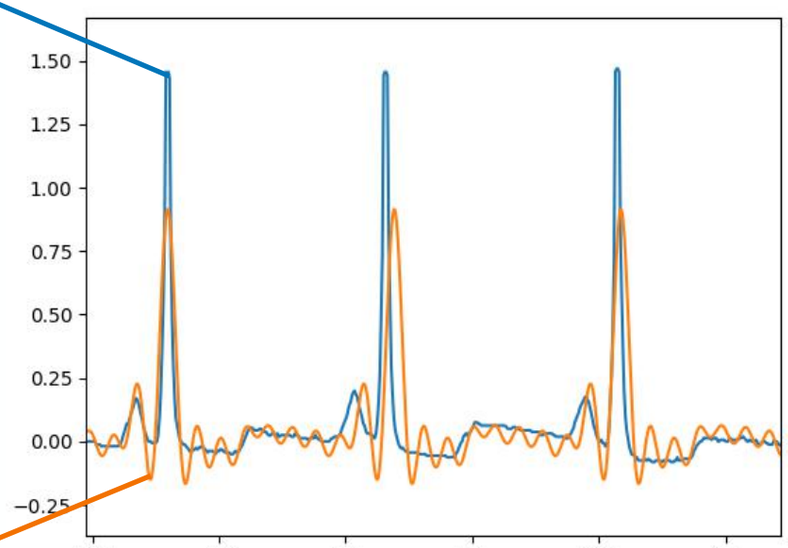
Lignes 332-340

Analyse Spectrale avec les Séries de Fourier

Résultats

Courbe Initiale

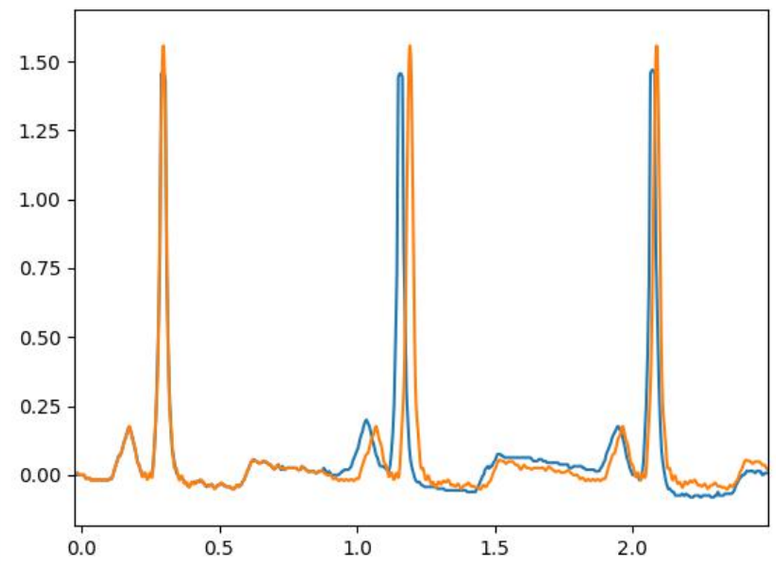


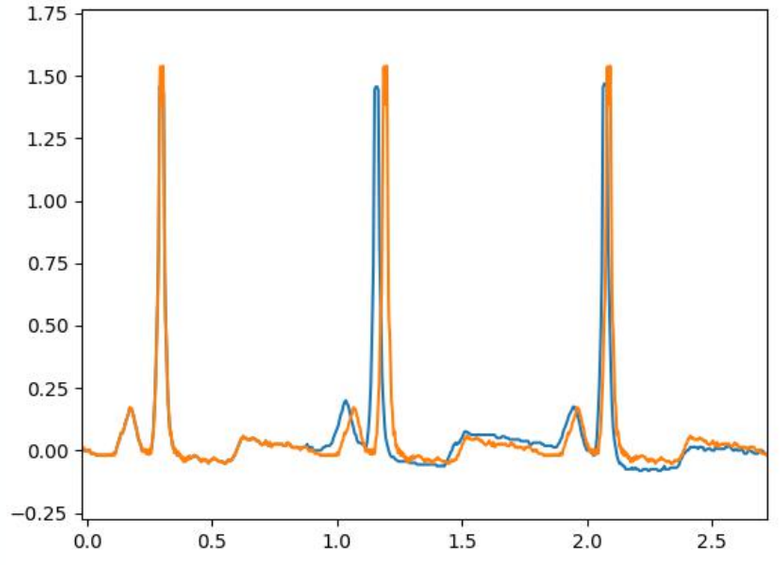


$S_5(f)$

Fonction ‘fonction_fourrier’ (ligne 286)

$S_{10}(f)$





$S_{30}(f)$

$S_{50}(f)$

Limites

- Dépendance avec la résolution de l’image
- Signal qui n’est pas exactement périodique
- Problèmes de détection des informations lors des ‘petits’ ECG

```
 68.61499364675984
Fréquence Cardiaque OK - Passez

 0.8744444444444445 Non Fiable
Rythme cardiaque anormal
Infarctus du Myocarde détecté
```

Annexe

Annexe

```python
import matplotlib.pyplot as plt
import matplotlib.image as img
import numpy as np
from math import cos
from math import sin
from math import pi

image1 = img.imread("file:///Users/florian_parfant/Documents/Khûbe/TIPE Khûbe/Base de Données ECG/ECG pour code/Normal Person ECG Images (284x12=3408)/Normal(118).png")

plt.imshow(image1)
plt.show()

#################
### Fonctions ###
#################

def x_premier_point(image):
    i,j=0,0
    while i<len(image[0]) and ((image[j][i][0]<0.3 or image[j][i][0]>0.7) and (image[j][i][1]<0.3 or image[j][i][1]>0.7) and (image[j][i][2]<0.3 or image[j][i][2]>0.7)):# or image[j][i][3]<0.1):
        while j<len(image) and ((image[j][i][0]<0.3 or image[j][i][0]>0.7) and (image[j][i][1]<0.3 or image[j][i][1]>0.7) and (image[j][i][2]<0.3 or image[j][i][2]>0.7)):# or image[j][i][3]<0.1):
            j+=1
        if j==len(image):
            j=0
            i+=1
    return i

def y_premier_point(image):
    i=x_premier_point(image)
    j=0
    k=1
    while j<len(image) and ((image[j][i+k][0]<0.3 or image[j][i+k][0]>0.7) and (image[j][i+k][1]<0.3 or image[j][i+k][1]>0.7) and (image[j][i+k][2]<0.3 or image[j][i+k][2]>0.7)):# or image[j][i+k][3]<0.1):
        while k<6 and ((image[j][i+k][0]<0.3 or image[j][i+k][0]>0.7) and (image[j][i+k][1]<0.3 or image[j][i+k][1]>0.7) and (image[j][i+k][2]<0.3 or image[j][i+k][2]>0.7)):# or image[j][i+k][3]<0.1):
            k+=1
        if k==6:
            k=1
            j+=1
    return j
```

Annexe

```python
def echelle(image):
    i,j=x_premier_point(image),y_premier_point(image)
    k=1
    while k<6 and ((image[j][i+k][0]<0.3 or image[j][i+k][0]>0.7) and (image[j][i+k][1]<0.3 or image[j][i+k][1]>0.7) and (image[j][i+k][2]<0.3 or image[j][i+k][2]>0.7)):# or image[j][i+k][3]<0.1):
        k+=1
    return k

def max_nombre(i,j):
    if i>j:
        return i
    else:
        return j

def max_global(liste):
    maxi=liste[0]
    for x in liste:
        if x>maxi:
            maxi=x
    return maxi

def min_global(liste):
    mini=liste[0]
    for x in liste:
        if x<mini:
            mini=x
    return mini

def max_local(liste,i1,i2):
    maxi=liste[i1]
    i=[i1]
    for x in range(i2-i1):
        if liste[i1+x]>maxi:
            maxi=liste[i1+x]
            i=[i1+x]
        elif liste[i1+x]==maxi:
            i.append(i1+x)
    return i,maxi

def min_local(liste,i1,i2):
    mini=liste[i1]
    i=[i1]
    for x in range(i2-i1):
        if liste[i1+x]<mini:
            mini=liste[i1+x]
            i=[i1+x]
        elif liste[i1+x]==mini:
            i.append(i1+x)
    return i,mini
```

Annexe

```python
def ecart_au_palier(i1,i2,tableau):
    (a,mini),(b,maxi) = min_local(tableau[1],i1,i2),max_local(tableau[1],i1,i2)
    return mini < min_global(tableau[1]) * 0.4 or maxi > max_global(tableau[1]) * 0.4

def reel_vers_pixel(x,liste):
    for i in range(len(liste)):
        if x==liste[i]:
            return i

def electro_tableau(image):
    largeur = len(image)
    longueur = len(image[0])
    tableau = np.zeros((2,longueur))
    for i in range(longueur):
        compt=0
        nbr=0
        for j in range(largeur):
            if image[j][i][0]<0.1 and image[j][i][1]<0.1 and image[j][i][2]<0.1:# and image[j][i][3]>0.1:
                compt+=j
                nbr+=1
        tableau[0][i]=i
        if compt==0:
            tableau[1][i]=tableau[1][i-1]
        else:
            tableau[1][i]=compt/nbr
    return tableau

def affinage_tableau(tableau,image):
    mini=min_global(tableau[1])
    maxi=max_global(tableau[1])
    a=max_nombre(palier-mini,maxi-palier)
    for i in range(len(tableau[0])):
        if tableau[1][i]<palier-a*0.7:
            minim=0
            j=0
            while minim==0:
                if image[j][i][0]<0.1 and image[j][i][1]<0.1 and image[j][i][2]<0.1:# and image[j][i][3]>0.1:
                    tableau[1][i]=j
                    minim=j
                j+=1
        if tableau[1][i]>palier+a*0.7:
            maxim=0
            for j in range(len(image)):
                if image[j][i][0]<0.1 and image[j][i][1]<0.1 and image[j][i][2]<0.1:# and image[j][i][3]>0.1:
                    maxim=j
            tableau[1][i]=maxim
    return tableau
```

Annexe

```python
def valeur_réel_tableau(tableau,image,echelle_abscisse,echelle_ordonnee,palier_image):
    k=echelle(image)
    x0=x_premier_point(image)
    x_echelle = echelle_abscisse/(10*k)
    y_echelle = echelle_ordonnee/(10*k)
    tab = np.zeros((2,len(tableau[0])))
    for i in range(len(tableau[0])):
        tab[0][i] = (i-x0)*x_echelle
    for j in range(len(tableau[1])):
        tab[1][j] = (palier-tableau[1][j])*y_echelle
    return tab

def pic_max(tableau,i0):
    i1,i2=i0,i0
    while i1<len(tableau[0]) and tableau[1][i1]<(max_global(tableau[1])*0.8):
        i1+=1
    i2=i1
    while i2<len(tableau[0]) and tableau[1][i2]>(max_global(tableau[1])*0.8):
        i2+=1
    if i1!=len(tableau[0]):
        i,maxi = max_local(tableau[1],i1,i2)
        somme=0
        for x in i:
            somme+=x
        return tableau[0][somme//len(i)],i1,i2
    else:
        return tableau[0][-1],i1,i2

def pic_min(tableau,i0):
    i1,i2=i0,i0
    while i1<len(tableau[0]) and tableau[1][i1]>(min_global(tableau[1])*0.8):
        i1+=1
    i2=i1
    while i2<len(tableau[0]) and tableau[1][i2]<(min_global(tableau[1])*0.8):
        i2+=1
    if i1!=len(tableau[0]):
        i,mini = min_local(tableau[1],i1,i2)
        somme=0
        for x in i:
            somme+=x
        return tableau[0][somme//len(i)],i1,i2
    else:
        return tableau[0][-1],i1,i2
```

Annexe

```python
def pic(tableau,i0):
    if abs(max_global(tableau[1]))>abs(min_global(tableau[1])):
        return pic_max(tableau,i0)
    else:
        return pic_min(tableau,i0)

def liste_pic(tableau):
    i0=0
    L=[]
    while i0<len(tableau[0]):
        abscisse_pic,i1,i2=pic(tableau,i0)
        if abscisse_pic!=tableau[0][-1]:
            i0=i2
            L.append(abscisse_pic)
        else:
            i0+=1
    return L

def distance_pic(liste,i1,i2):
    return (liste[i2]-liste[i1])/(i2-i1)

def liste_pic_pixel(tableau):
    liste=liste_pic(tableau)
    res=[]
    for x in liste:
        res.append(reel_vers_pixel(x,tableau[0]))
    return res

def fréquence(tableau):
    i0=0
    L=[]
    compt=0
    while i0<len(tableau[0]):
        abscisse_pic,i1,i2=pic(tableau,i0)
        if abscisse_pic!=tableau[0][-1]:
            compt+=1
            i0=i2
            L.append(abscisse_pic)
        else:
            i0+=1
    abs_premier_pic=L[0]
    abs_dernier_pic=L[len(L)-1]
    print(L)
    return ((compt-1)/(abs_dernier_pic-abs_premier_pic))*60
```

Annexe

```python
def rythme_cardiaque(tableau):
    L,n=liste_pic(tableau),len(liste_pic(tableau))
    return distance_pic(L,0,n-1)

def fiabilite_RC(tableau):
    pres = []
    L,n=liste_pic(tableau),len(liste_pic(tableau))
    RC_theorique = rythme_cardiaque(tableau)
    for i in range(n-1):
        pres.append(distance_pic(L,i,i+1))
        if distance_pic(L,i,i+1) > 1.1 * RC_theorique or distance_pic(L,i,i+1) < 0.9 * RC_theorique:
            return "Non Fiable"
    print(pres)
    return "Fiable"

def anomalie_myocarde(tableau):
    l_pic = liste_pic_pixel(tableau)
    for i in range(len(l_pic)-1):
        taille_pic = abs(tableau[1][l_pic[i]])
        ecart = l_pic[i+1]-l_pic[i]
        i1,i2=int(l_pic[i]+ecart/8),int(l_pic[i+1]-ecart/4)
        for x in tableau[1][i1:i2]:
            if x > taille_pic*0.35:
                plt.plot(tableau[0][:int(l_pic[i+1])],tableau[1][:int(l_pic[i+1])])
                plt.plot(tableau[0][i1:i2],tableau[1][i1:i2])
                plt.axis('equal')
                plt.show()
                return True
    return False

def retrouver_indice(liste,x):
    n=len(liste)
    for i in range(n-1):
        if liste[i]<=x and liste[i+1]>x:
            if abs(liste[i]-x)<=abs(liste[i+1]-x):
                return i
            else:
                return i+1
```

Annexe

```
264 def fonction_fourier_cos(tableau,n,période):
265     tab = np.zeros((2,len(tableau[0])))
266     for i in range(len(tableau[0])):
267         tab[0][i]=tableau[0][i]
268     for j in range(len(tableau[1])):
269         tab[1][j]=(tableau[1][j])*cos(n*tableau[0][j]*(2*pi)/période)
270     return tab
271 
272 def fonction_fourier_sin(tableau,n,période):
273     tab = np.zeros((2,len(tableau[0])))
274     for i in range(len(tableau[0])):
275         tab[0][i]=tableau[0][i]
276     for j in range(len(tableau[1])):
277         tab[1][j]=(tableau[1][j])*sin(n*tableau[0][j]*(2*pi)/période)
278     return tab
279 
280 def trapèze(tableau,a,ind_periode):
281     somme=0
282     for i in range(ind_periode):
283         somme+=((tableau[1][a+i]+tableau[1][a+i+1])/2)*(tableau[0][a+i+1]-tableau[0][a+i])
284     return somme
285 
286 def fonction_fourier(liste_a,liste_b,x,période):
287     somme=0
288     for i in range(len(liste_a)):
289         somme+=liste_a[i]*cos(i*x*(2*pi)/période)+liste_b[i]*sin(i*x*(2*pi)/période)
290     return somme
```

Annexe

```
292 ##################
293 ### Programme ###
294 ##################
295 
296 t = electro_tableau(image1)
297 palier = t[1][0]
298 
299 X0=t[0]
300 Y0=t[1]
301 Ypal=[palier]*len(t[1])
302 
303 plt.imshow(image1)
304 plt.plot(X0,Y0)
305 plt.plot(X0,Ypal)
306 plt.show()
307 
308 t_affiné = affinage_tableau(t,image1)
309 valeur_reel = valeur_réel_tableau(t_affiné,image1,0.2,0.5,palier)
310 FC = fréquence(valeur_reel)
311 Rythme = rythme_cardiaque(valeur_reel)
312 Fiabilite = fiabilite_RC(valeur_reel)
313 
314 X=t_affiné[0]
315 Y=t_affiné[1]
316 
317 plt.imshow(image1)
318 plt.plot(X,Y)
319 plt.plot(X,Ypal)
320 plt.show()
321 
322 X1=valeur_reel[0]
323 Y1=valeur_reel[1]
324 
325 plt.plot(X1,Y1)
326 plt.axis('equal')
327 plt.show()
328 
329 indice_periode = retrouver_indice(valeur_reel[0],rythme_cardiaque(valeur_reel))
330 periode = valeur_reel[0][indice_periode]-valeur_reel[0][0]
```

Annexe

```
coeff_a = [(1/periode)*trapèze(fonction_fourier_cos(valeur_reel,0,periode),0,indice_periode)]
coeff_b = [0]

for i in range(1,50):
    coeff_a.append((2/periode)*trapèze(fonction_fourier_cos(valeur_reel,i,periode),0,indice_periode))
    coeff_b.append((2/periode)*trapèze(fonction_fourier_sin(valeur_reel,i,periode),0,indice_periode))

print(coeff_a)
print(coeff_b)

Y_fourier=[]
n=20000
#for x in X1:
#    Y_fourier.append(fonction_fourier(coeff_a,coeff_b,x,periode))
for x in [X1[0]+i*((X1[-1]-X1[0])/(n-1)) for i in range(n)]:
    Y_fourier.append(fonction_fourier(coeff_a,coeff_b,x,periode))

plt.plot(X1,Y1)
plt.plot([X1[0]+i*((X1[-1]-X1[0])/(n-1)) for i in range(n)],Y_fourier)
#plt.plot(X1,Y_fourier)
plt.axis('equal')
plt.show()

print("\n",echelle(image1))

print("\n",FC)

if FC < 50:
    print("Bradycardie - Consultez vite un Cardiologue :'(")
elif FC >= 50 and FC < 60:
    print("Suspicion de Bradycardie - Prise de Rendez-vous chez un Cardiologue vivement conseillé :/")
elif FC >= 60 and FC <= 90:
    print("Fréquence Cardiaque OK - Passez une Bonne Journée :)")
elif FC > 90 and FC <= 100:
    print("Suspicion de Tachycardie - Prise de Rendez-vous chez un Cardiologue vivement conseillé :/")
else:
    print("Tachycardie - Consultez vite un Cardiologue :'(")

print("\n",Rythme,Fiabilite)

if Fiabilite == "Fiable":
    print("Rythme cardiaque régulier")
else:
    print("Rythme cardiaque anormal")

if anomalie_myocarde(valeur_reel):
    print("Infarctus du Myocarde détecté")
else:
    print("Pas d'Infarctus du Myocarde détecté")
```