Détection de feux de forêts

Objectifs du TIPE

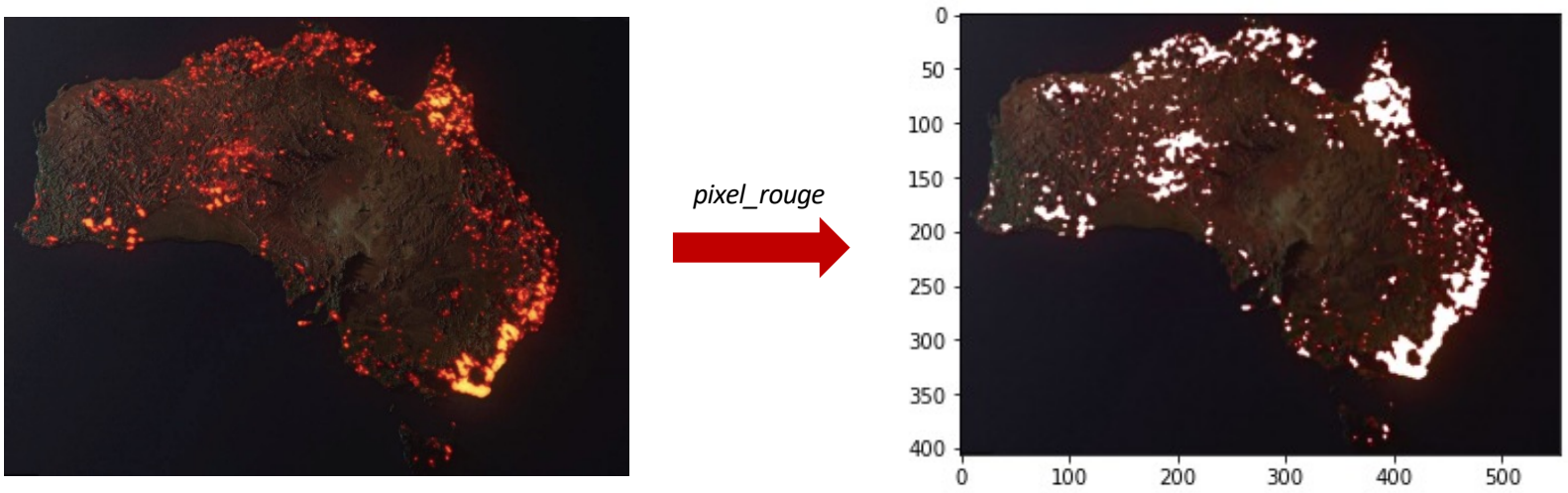
- Déterminer une méthode efficace de détection d'un feu de forêt
- Ajuster cette méthode
- Choisir le type d'images le mieux adapté pour cette détection
- Approximer la taille du feu

Sommaire

1. Présentation de la méthode
2. Critiques et améliorations à apporter
3. Image thermique ou photographie
4. Approximation de la taille du feu

- Principe de l'algorithme

Détection grâce aux pixels rouges



- Principe de l'algorithme

Quand considère-t-on un pixel rouge ?

Composante R > 0,8

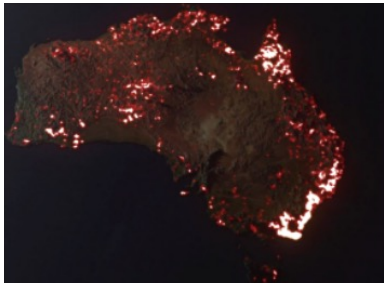
Présentation de la méthode de détection

- Conditions de déclenchement de l'algorithme

Taux de pixels rouges sur les images de feu

pixel_rouge

N=7090
T=589824
1,2%



pixel_rouge

N=3441
T=224924
1,5%

pixel_rouge

N=55026
T=1112320
4,9%

- Conditions de déclenchement de l'algorithme

Taux de pixels rouges sur les images de feu

pixel_rouge

N=9
T=600000
0,0015%

Présentation de la méthode de détection

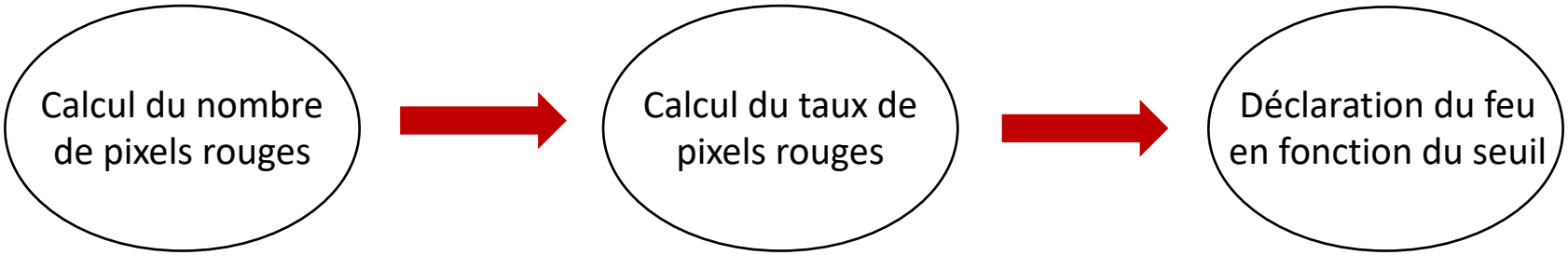
- Conditions de déclenchement de l'algorithme

Seuil de pixels rouges

Taux de pixels rouges
supérieur à **1%**

- Fonctionnement de l'algorithme

Premier programme *feu1*



- Fonctionnement de l'algorithme

Premier programme *feu1*

feu1

'Feu'

feu1

'Feu'

feu1

'Feu'

feu1

'Pas de feu'

Présentation de la méthode de détection

- Limite de l'algorithme

feu1 → 'Feu'

Présentation de la méthode de détection

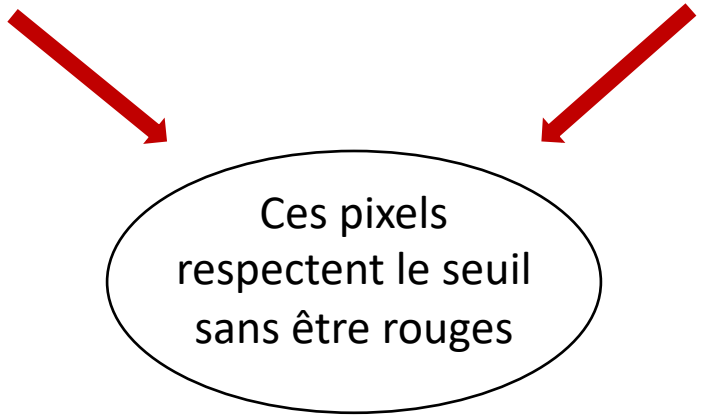
Causes → Seuil rouge uniquement

Pixel gris choisi dans les nuages :

R: 0.85
V: 0.88
B: 0.9

Pixel orange choisi dans les feuilles :

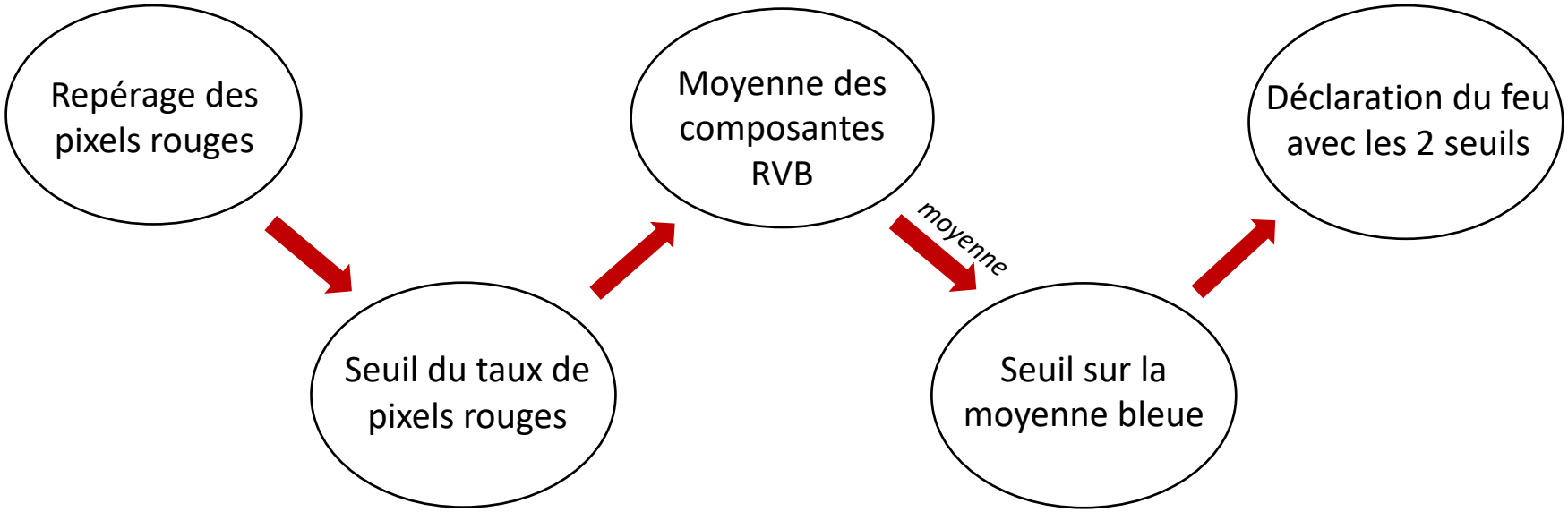
R: 0.95
V: 0.57
B: 0.24



- Limite de l'algorithme

Deuxième programme *feu2*

Solution → Ajout d'un seuil sur les composantes BLEUES des pixels rouges



- Limite de l'algorithme

Deuxième programme *feu2*

feu2

'Feu'

feu2

'Feu'

feu2

'Feu'

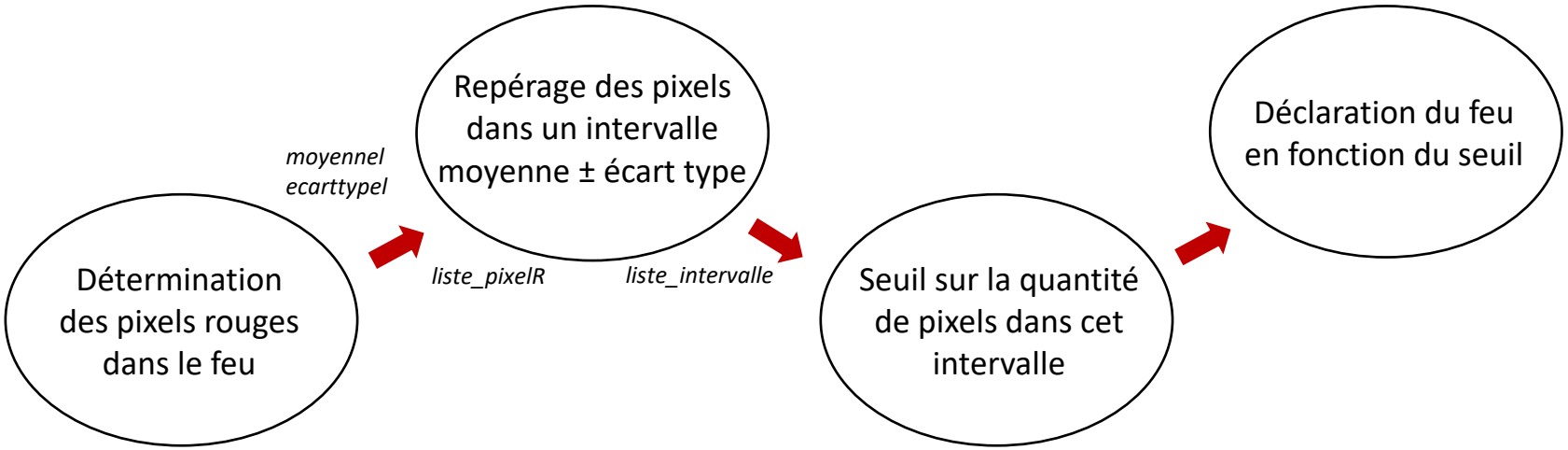
feu2

'Pas de feu'

- Nouvelle méthode

Méthode → Statistiques sur les valeurs des composantes des pixels

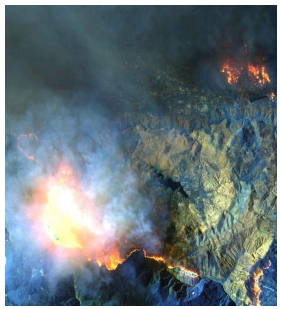
Programme *statistiques*



Présentation de la méthode de détection

- Nouvelle méthode





statistiques *statistiques* *statistiques* *statistiques* *statistiques*

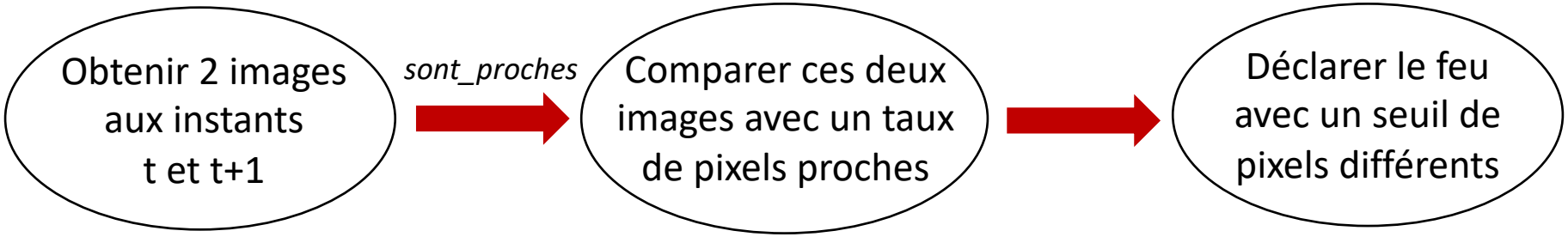
'Feu' 'Feu' 'Feu' 'Pas de feu' 'Pas de feu'

Critiques et améliorations à apporter

- Confusion entre un feu et un paysage rouge

Méthode → Le feu est mobile, contrairement à un paysage rouge

Programme *mouvement*



Critiques et améliorations à apporter

img1 à l'instant t

img2 à l'instant t

mouvement

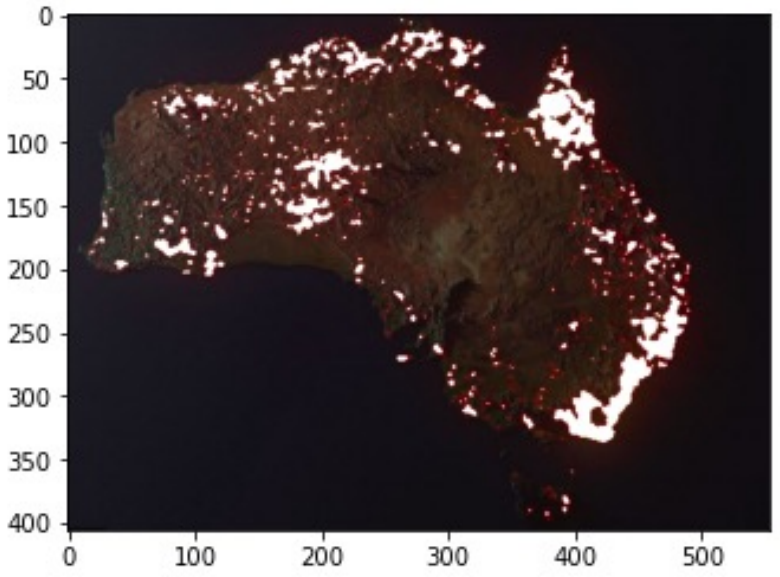
'Pas de feu'

mouvement

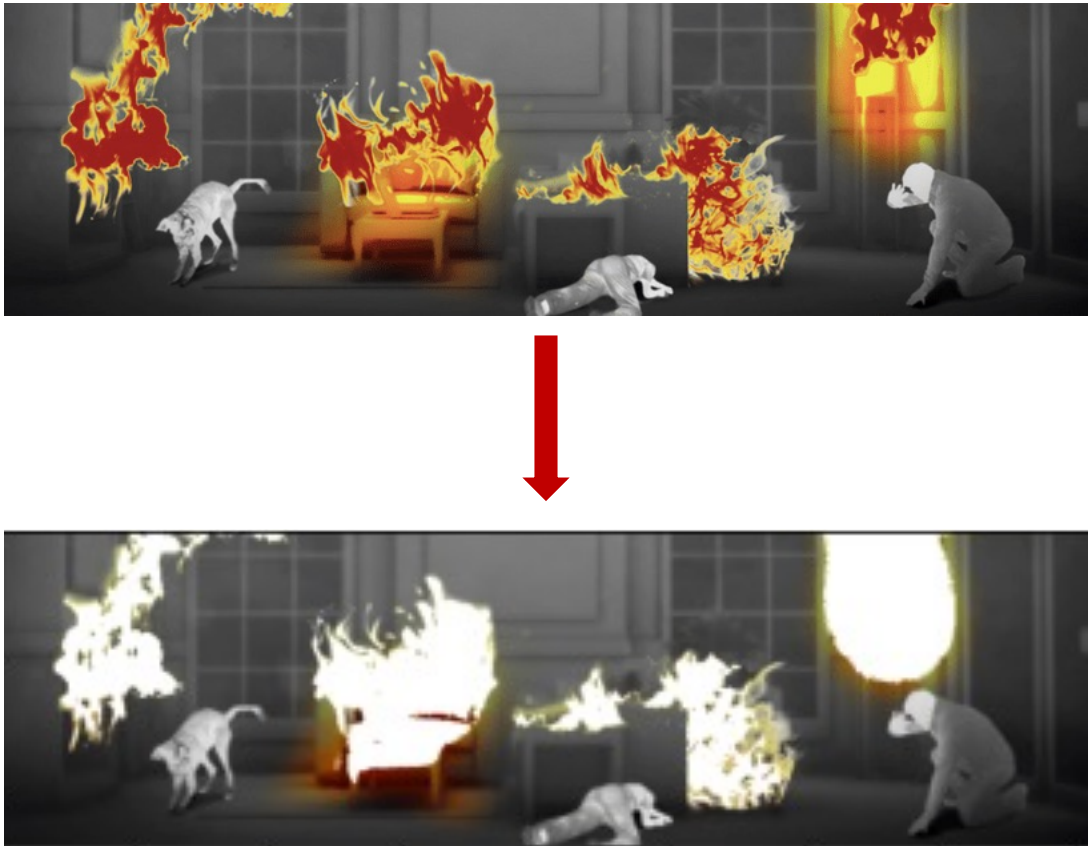
'Feu'

- Analyse de la méthode *feu2* sur une photographie

Quelques feux ponctuels non repérés



- Analyse de la méthode *feu2* sur une image thermique



Tous les feux et uniquement les feux sont repérés

Image thermique ou photographie

Choix retenu → Image thermique

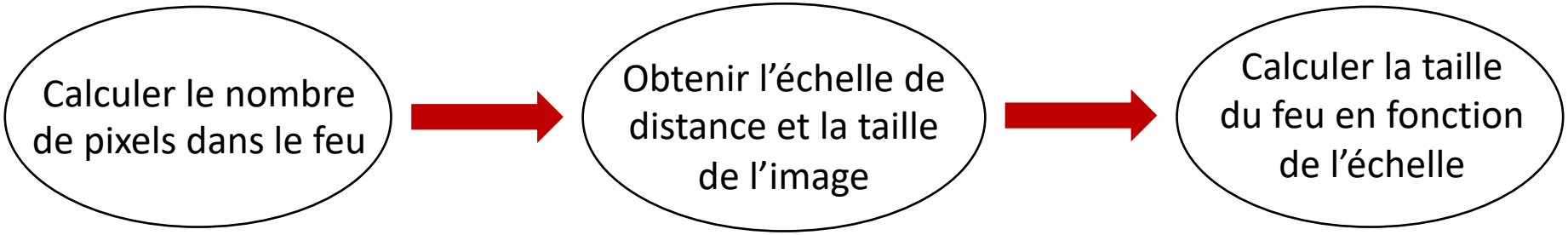
Pourquoi l'image thermique est-elle plus efficace ? → Meilleur contraste entre les couleurs

Limite → Images plus difficiles à obtenir

- Approximation de la taille du feu

Méthode → Connaître la superficie du territoire étudié sur l'image

Programme *taille*



Approximation de la taille du feu

- Exemple de l'Australie

Superficie du territoire : 7,692 millions km²
Taille de l'image : programme *pixels_terre*

Remplacer le fond marin par des pixels blancs

↓

Repérer les pixels non blancs de l'image

↓

Calculer ainsi la taille du territoire

Approximation de la taille du feu

- Exemple de l'Australie

Superficie du territoire : 7,692 millions km²
Taille de l'image : 1 737 604 pixels

1 pixel = 4,43 km²

Résultat du programme *taille :* **308 179.5 km²**

Codes

```
def pixel_rouge(img):
    (u,v,w)=img.shape
    newimg=np.zeros((u,v,3))
    c=0
    for i in range (u):
        for j in range (v):
            if img[i][j][0]>=0.8:
                newimg[i][j]=[1,1,1]
                c=c+1
            else : newimg[i][j]=[img[i][j][0],img[i][j][1],img[i][j][2]]
    plt.imshow(newimg)
    return c
```

Codes

```
def nb_pixelR(img):
    (u,v,w)=img.shape
    c=0
    for i in range (u):
        for j in range (v):
            if img[i][j][0]>=0.8:
                c=c+1
    return c
```

```
def feu1(img):
    nb=nb_pixelR(img)
    taille=img.shape[0]*img.shape[1]
    seuil=100*nb/taille
    if seuil<=1:
        return ("Pas de feu")
    else :
        return ("Feu")
```

Codes

```
def moyenne(img):
    (u,v,w)=img.shape
    sR=0
    sV=0
    sB=0
    for i in range(u):
        for j in range (v):
            sR=sR+img[i][j][0]
            sV=sV+img[i][j][1]
            sB=sB+img[i][j][2]
    mR=sR/(u*v)
    mV=sV/(u*v)
    mB=sB/(u*v)
    return [mR,mV,mB]
```

```
def feu2(img):
    R=nb_pixelR(img)
    V=moyenne(img)[1]
    B=moyenne(img)[2]
    taille=img.shape[0]*img.shape[1]
    seuilR=100*R/taille
    if seuilR<=1:
        return ("Pas de feu")
    elif B>=0.5 :
        return ("Pas de feu")
    else :
        return ("Feu")
```

Codes

```
def ecarttypel(l):
    mr=moyennel(l)[0]
    mb=moyennel(l)[1]
    vr=0
    vb=0
    for x in l :
        vr=vr+(x[0]-mr)**2
        vb=vb+(x[2]-mb)**2
    return[sqrt(vr),sqrt(vb)]
```

```
def moyennel(l):
    n=len(l)
    r=0
    b=0
    for x in l :
        r=r+x[0]
        b=b+x[2]
    return[r/n,b/n]
```

Codes

```
def liste_pixelR(img):
    (u,v,w)=img.shape
    l=[]
    for i in range (u):
        for j in range (v):
            if img[i][j][0]>=0.8 and img[i][j][2]<=0.5:
                l.append(img[i][j])
    return l
```

```
def statistiques(img):
    c=liste_intervalle(img)
    (u,v,w)=img.shape
    taille=u*v
    seuil=100*c/taille
    if seuil<=1:
        return ('Pas de feu')
    else :
        return ('Feu')
```

Codes

```
def sont_proches(p1,p2,e):
    if (abs(p1[0]-p2[0])<e) and (abs(p1[1]-p2[1])<e) and (abs(p1[2]-p2[2])<e):
        return True
    else :
        return False


def mouvement(img1,img2):
(u,v,w)=img1.shape
taille=u*v
c=0
for i in range(u):
    for j in range(v):
        if sont_proches(img1[i][j],img2[i][j],0.05):
            c=c+1
taux=100*c/taille
if taux<=90:
    return("Feu")
else:
    return("Pas de feu")
```

Codes

```
def pixels_terre(img):
    (u,v,w)=img.shape
    c=0
    for i in range(u):
        for j in range(v):
            if img[i][j][0]!=1 and img[i][j][1]!=1 and img[i][j][2]!=1:
                c=c+1
    return c


def nb_pixelRouge(img):
    (u,v,w)=img.shape
    c=0
    for i in range (u):
        for j in range (v):
            if img[i][j][0]>=0.8 and img[i][j][2]<=0.5:
                c=c+1
    return c
```

Codes

```
def taille(img,superficie):
    r=nb_pixelRouge(img)
    t=pixels_terre(img)
    echelle=superficie/t
    feu=r*echelle
    return feu
```