NAVIGATION ASTRONOMIQUE :
SE REPÉRER GRÂCE À UNE PHOTO DU CIEL

29488
NIELS GUÉGAN-LECLERC

Source Image : https://blog.loisirsplaisirs.com/stage-astronomie/

INTRODUCTION

Navigation astronomique : technique historique de navigation permettant de se repérer sur Terre grâce à l'observation des astres.

Depuis l'arrivée des technologies modernes, cette pratique a été délaissée au profit de navigation GPS.

Le sextant

Source Image : https://www.futura-sciences.com/sciences/questions-reponses/espace-sextant-principe-utilisation-1616/

PROBLÉMATIQUE

Dans quelle mesure l'analyse des étoiles présentes sur une photo peut-elle renseigner sur la position d'un observateur sur Terre ?

Source Image : https://www.polarys.com/letoile-polaire-polaris/

PLAN

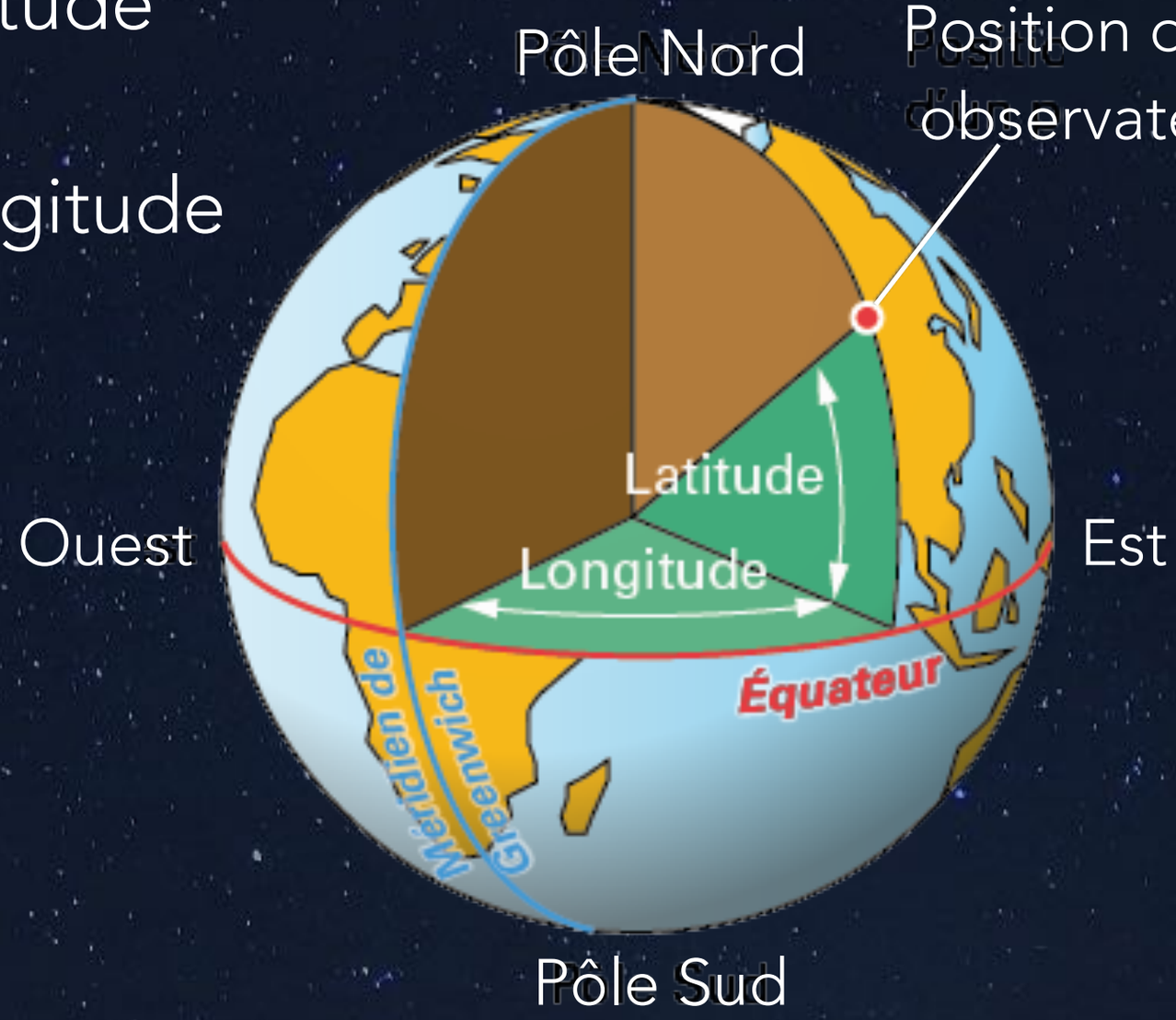
- **En théorie :** comment se servir des astres comme point de repère
- **En pratique :** trouver sa position sur Terre à partir de photographies adaptées
- Vers un **programme informatique** permettant de retrouver la position d'un observateur automatiquement

EN THÉORIE : COMMENT SE SERVIR DES ASTRES COMME POINT DE REPÈRE

Trouver la position d'un observateur sur Terre c'est :

- Déterminer sa latitude

- Déterminer sa longitude



Source Image : https://www.digischool.fr/cours/le-reperage-a-la-surface-de-la-terre

DÉTERMINATION DE LA LATITUDE

L'ÉTOILE POLAIRE

On détermine la latitude à partir de l'étoile polaire.

Celle-ci a la particularité de se situer dans le prolongement de l'axe de rotation de la Terre.

Conséquence de cette particularité : la position de l'étoile polaire dans le ciel ne dépend pas du temps mais uniquement de l'endroit sur Terre duquel on l'observe.

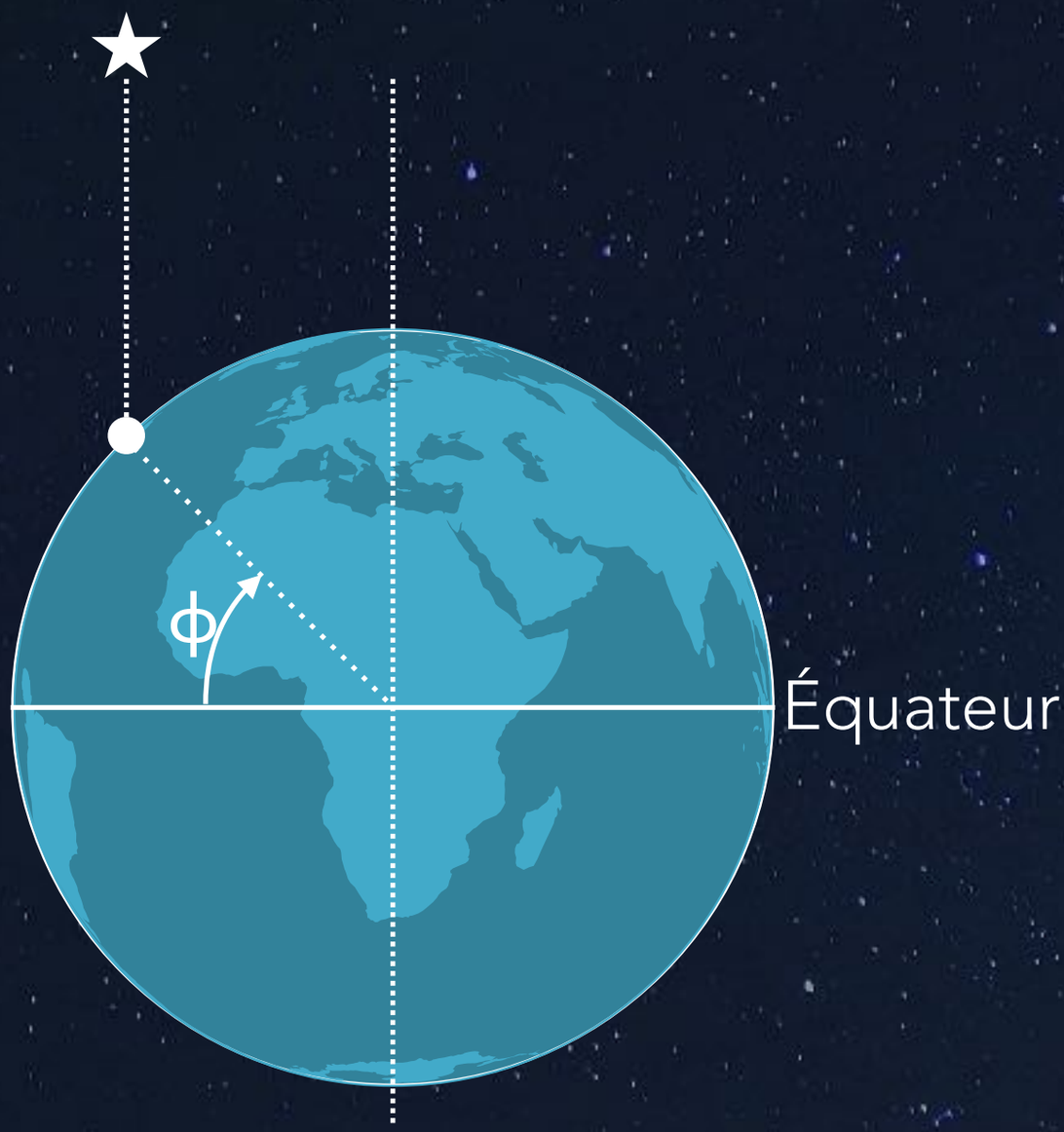
Filé d'étoiles

Source Image : https://fr.freepik.com/photos-premium/trainees-etoiles-dans-ciel-nocturne-vue-espace-etoile-lumiere-au-dessus-ligne-horizon_15225363.htm

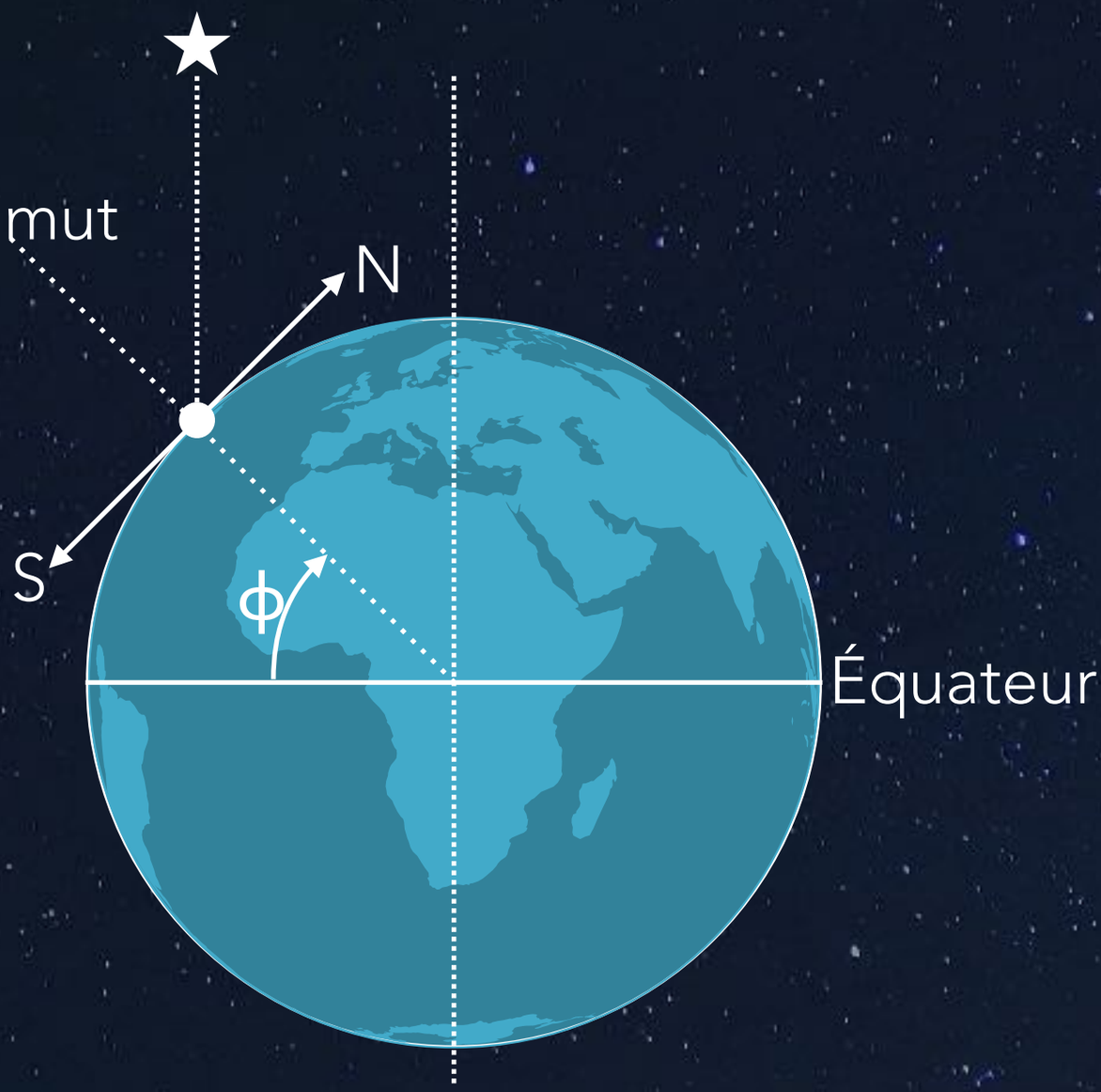
LIEN AVEC LA LATITUDE DE L'OBSERVATEUR

LIEN AVEC LA LATITUDE DE L'OBSERVATEUR

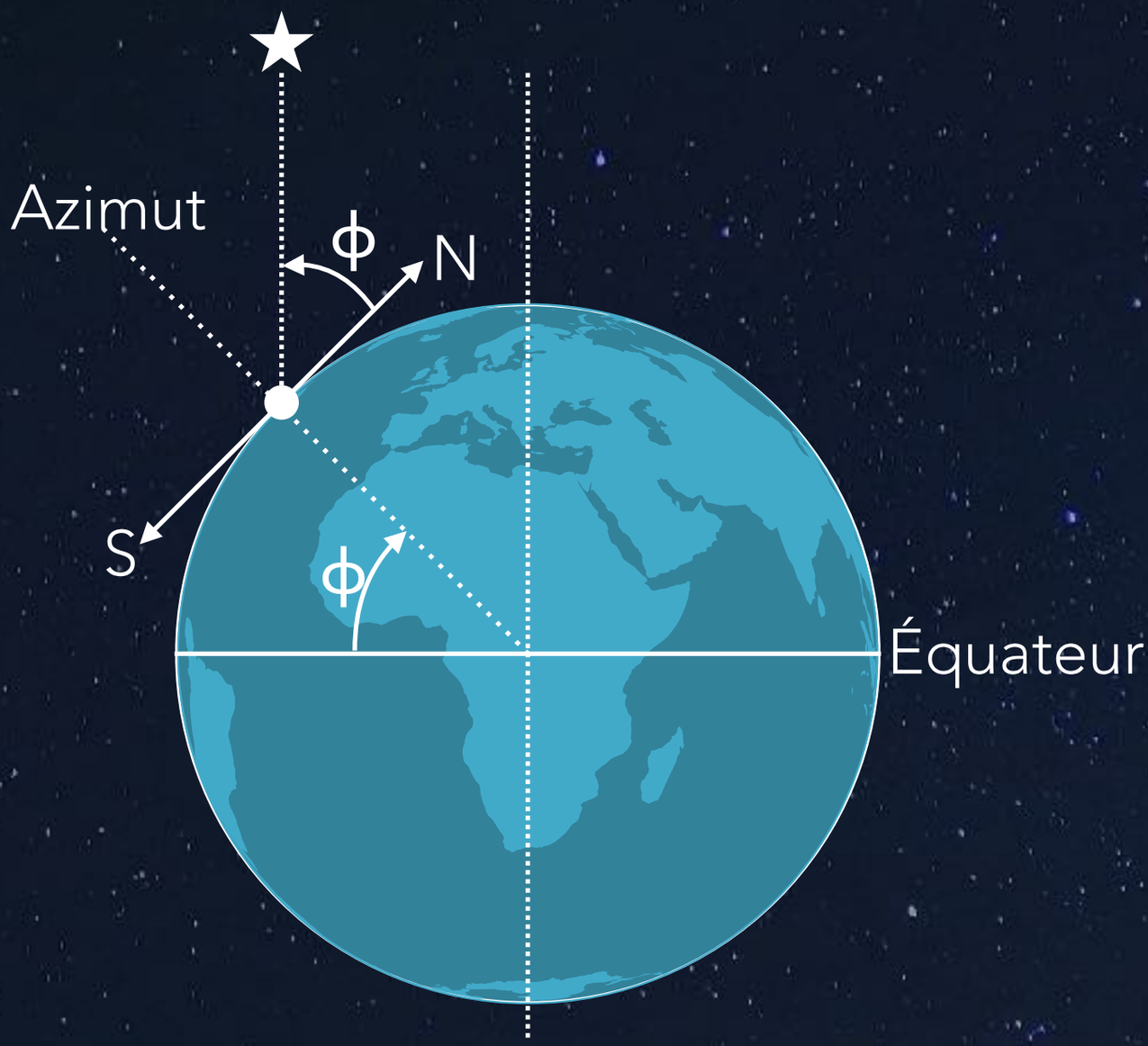
LIEN AVEC LA LATITUDE DE L'OBSERVATEUR



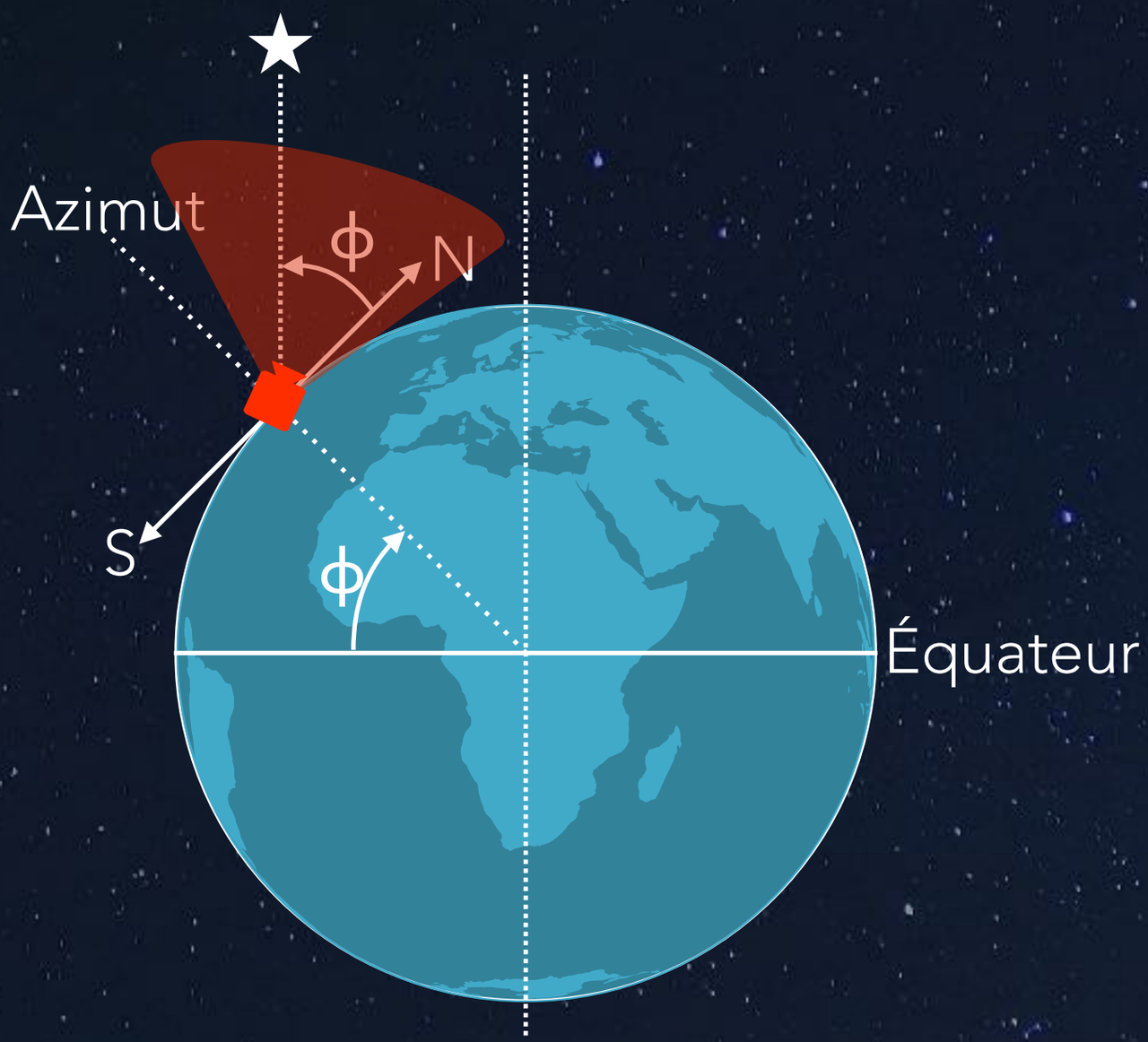
LIEN AVEC LA LATITUDE DE L'OBSERVATEUR



LIEN AVEC LA LATITUDE DE L'OBSERVATEUR

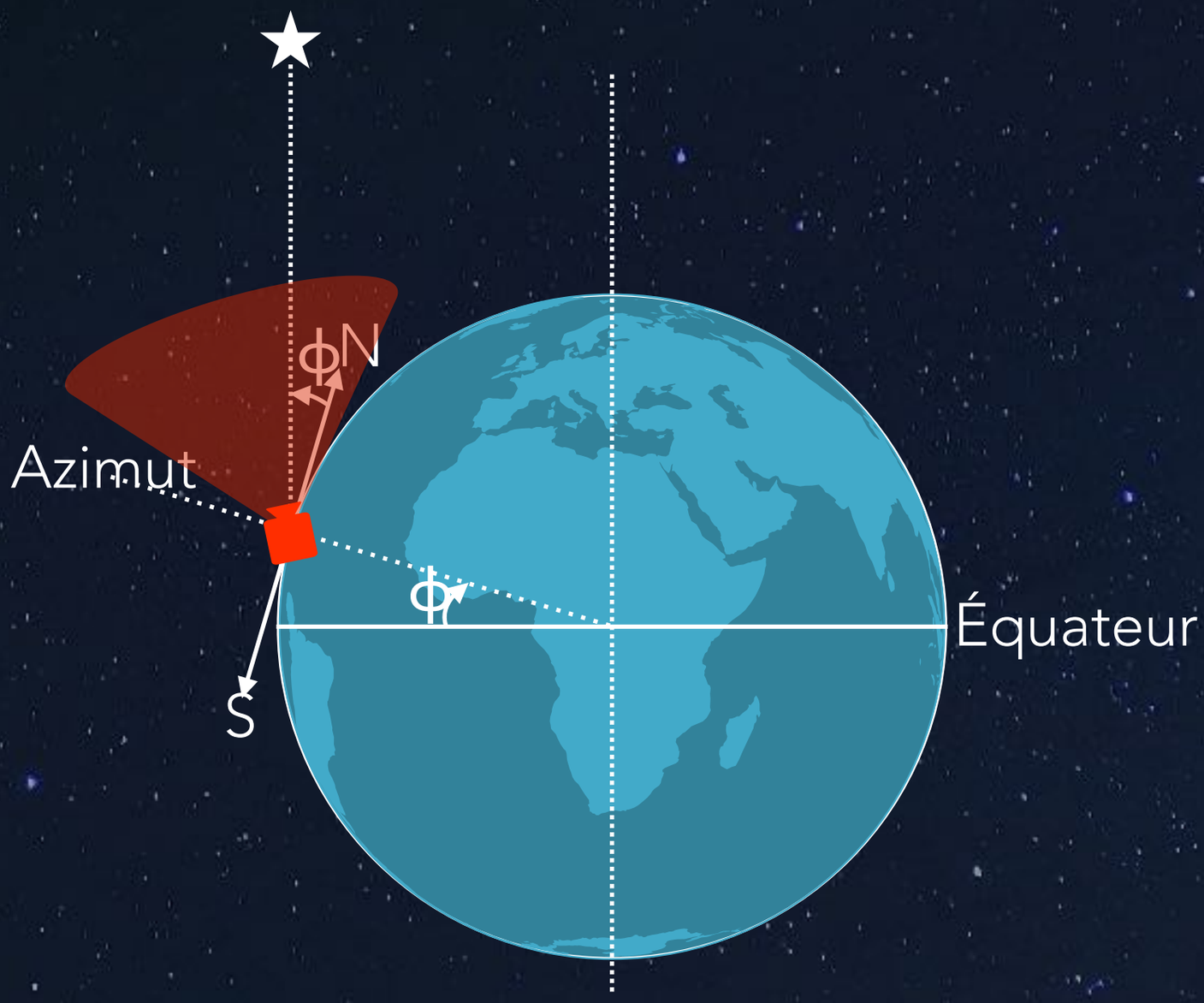


LIEN AVEC LA LATITUDE DE L'OBSERVATEUR



(En astronomie, ***la hauteur*** *d'un astre correspond à*
l'angle formé entre cet astre et l'horizon*)*

LIEN AVEC LA LATITUDE DE L'OBSERVATEUR



Conclusion, pour retrouver la latitude de l'observateur, on propose alors le protocole suivant :

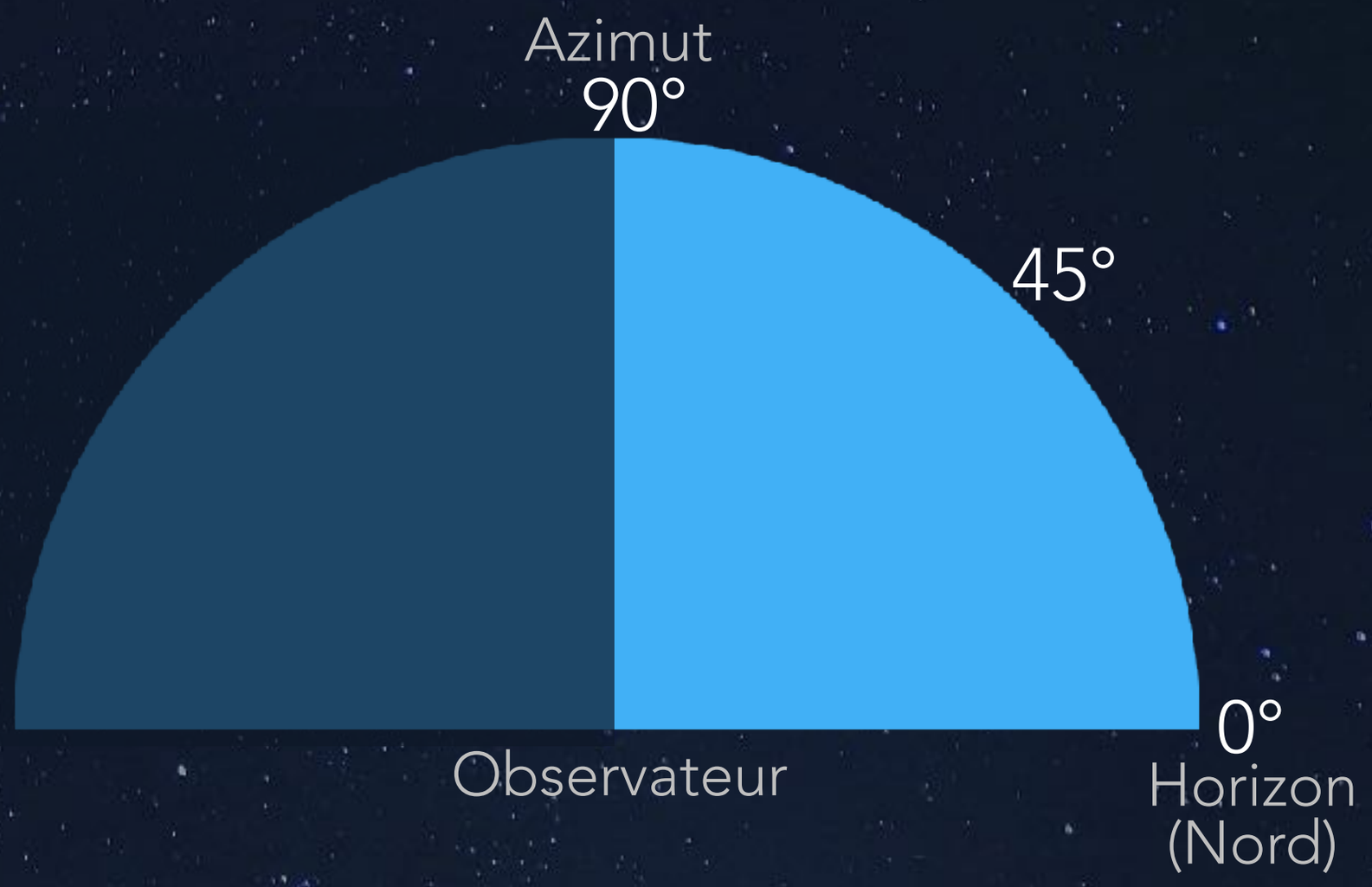
- **On prend une photo du ciel vers le Nord** avec une ouverture angulaire capable de couvrir des angles de 0° (horizon) à 90° (azimut).

- **On mesure la distance entre l'étoile polaire** (nous verrons comment la repérer sur la photo par la suite) **et l'horizon**. Cette distance nous renseigne alors sur la latitude de l'observateur.

LIMITES DE CE PROTOCOLE

1) La plupart des smartphones ne disposent pas d'une résolution angulaire suffisante.

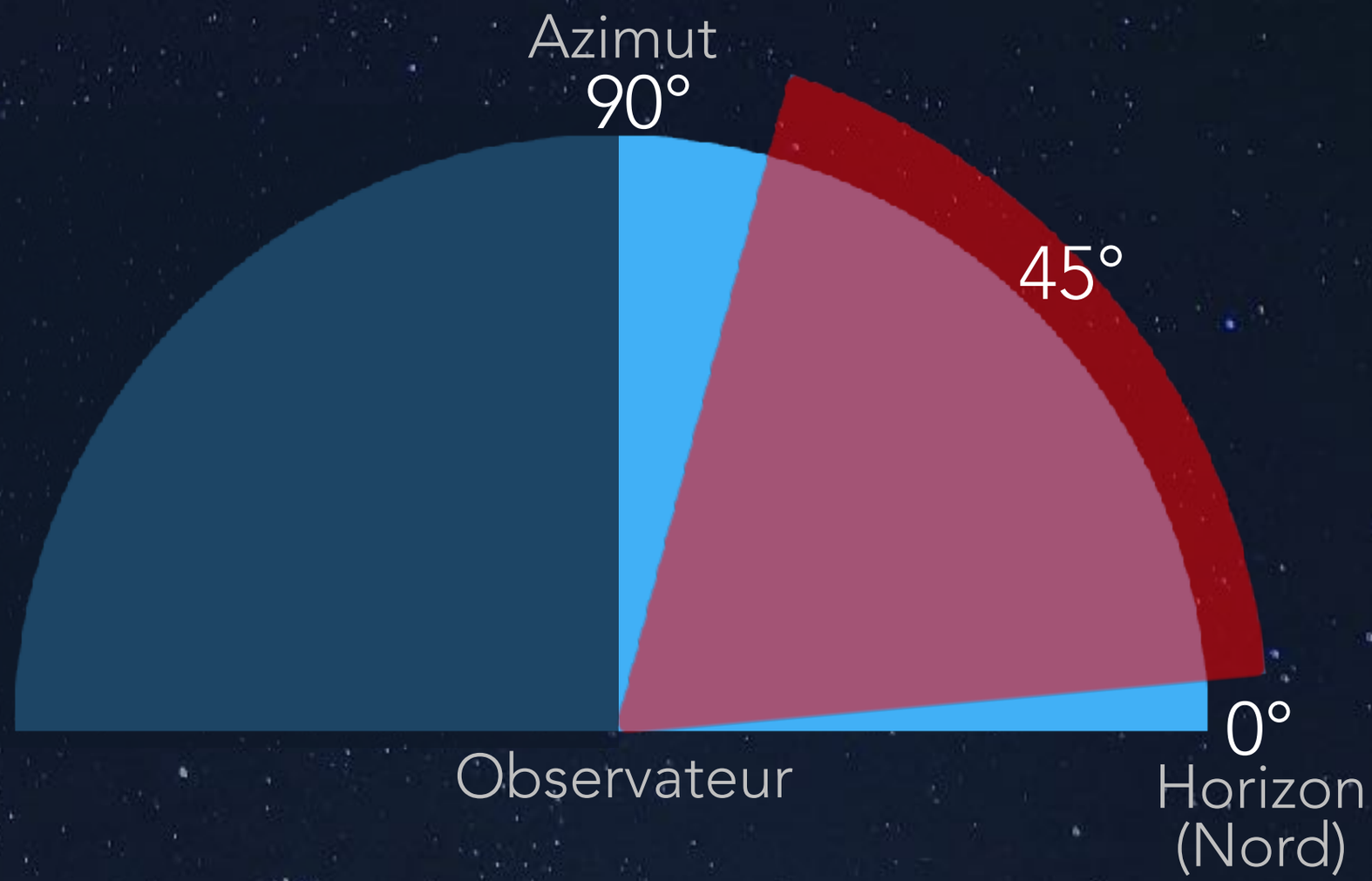
→ solution : on segmente en 3 photos respectivement réalisées avec des angles de 0° (horizon), 45°, 90° (azimut).



LIMITES DE CE PROTOCOLE

1) La plupart des smartphones ne disposent pas d'une résolution angulaire suffisante.

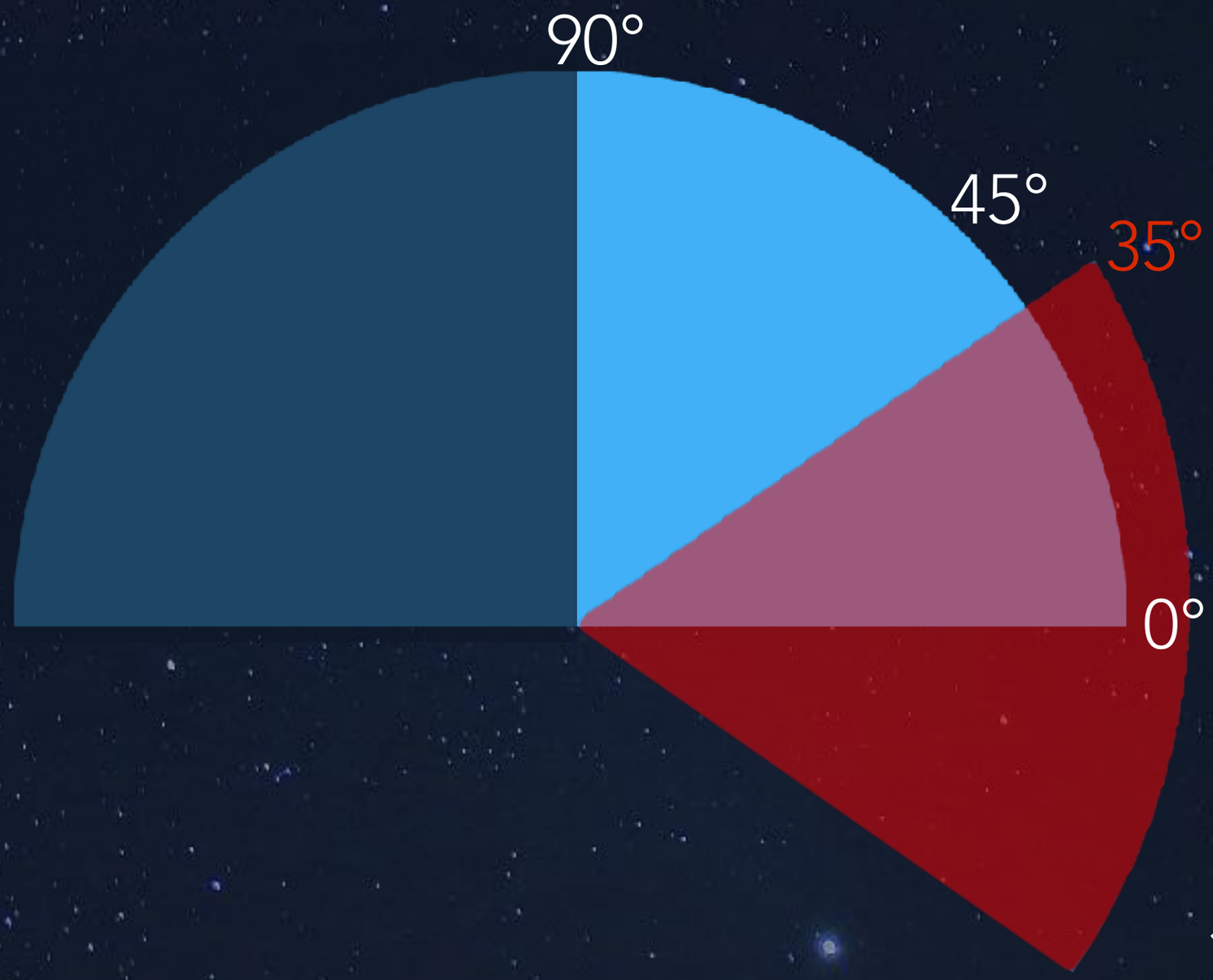
→ solution : on segmente en 3 photos respectivement réalisées avec des angles de 0° (horizon), 45°, 90° (azimut).



LIMITES DE CE PROTOCOLE

1) La plupart des smartphones ne disposent pas d'une résolution angulaire suffisante.

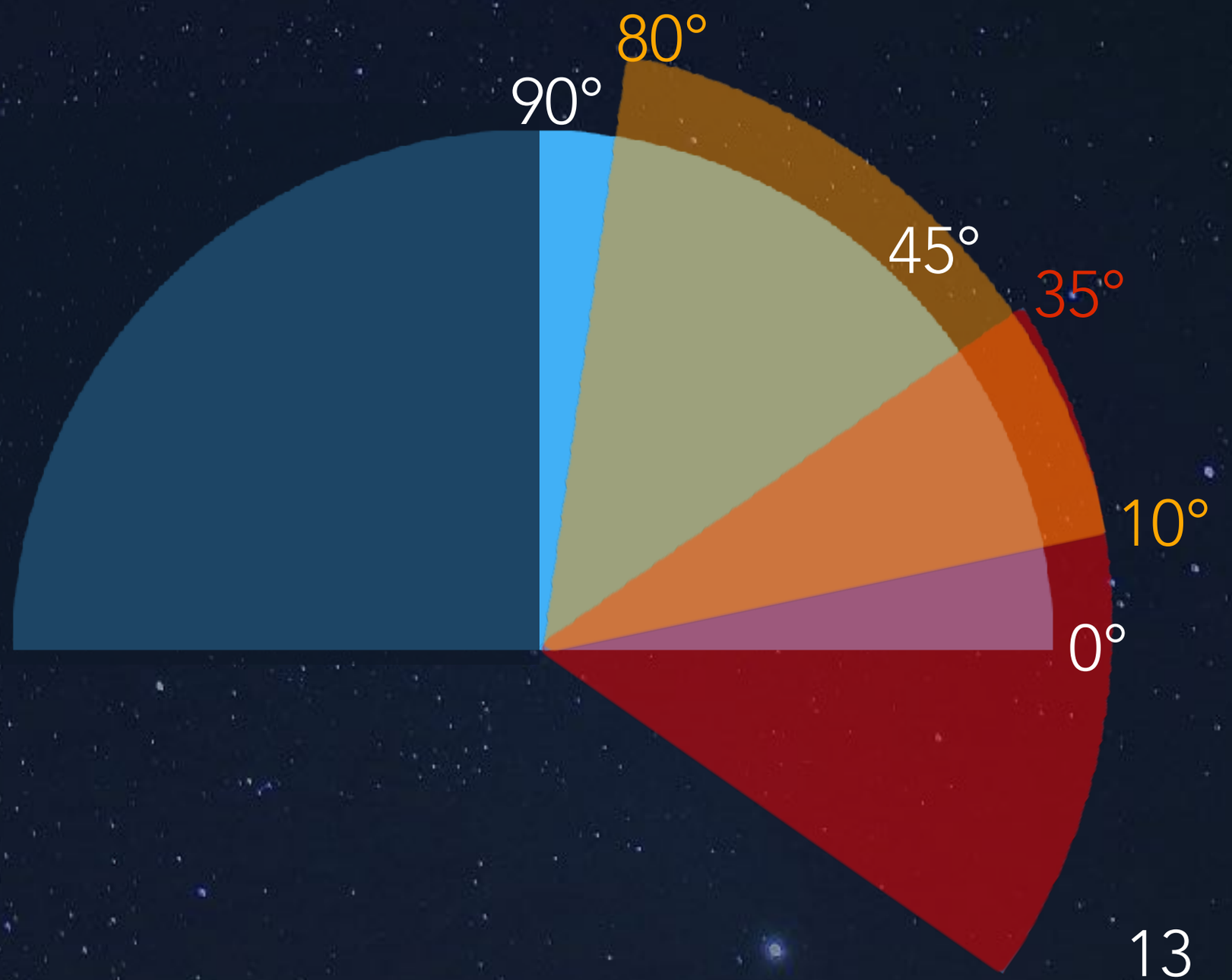
→ solution : on segmente en 3 photos respectivement réalisées avec des angles de 0° (horizon), 45°, 90° (azimut).



LIMITES DE CE PROTOCOLE

1) La plupart des smartphones ne disposent pas d'une résolution angulaire suffisante.

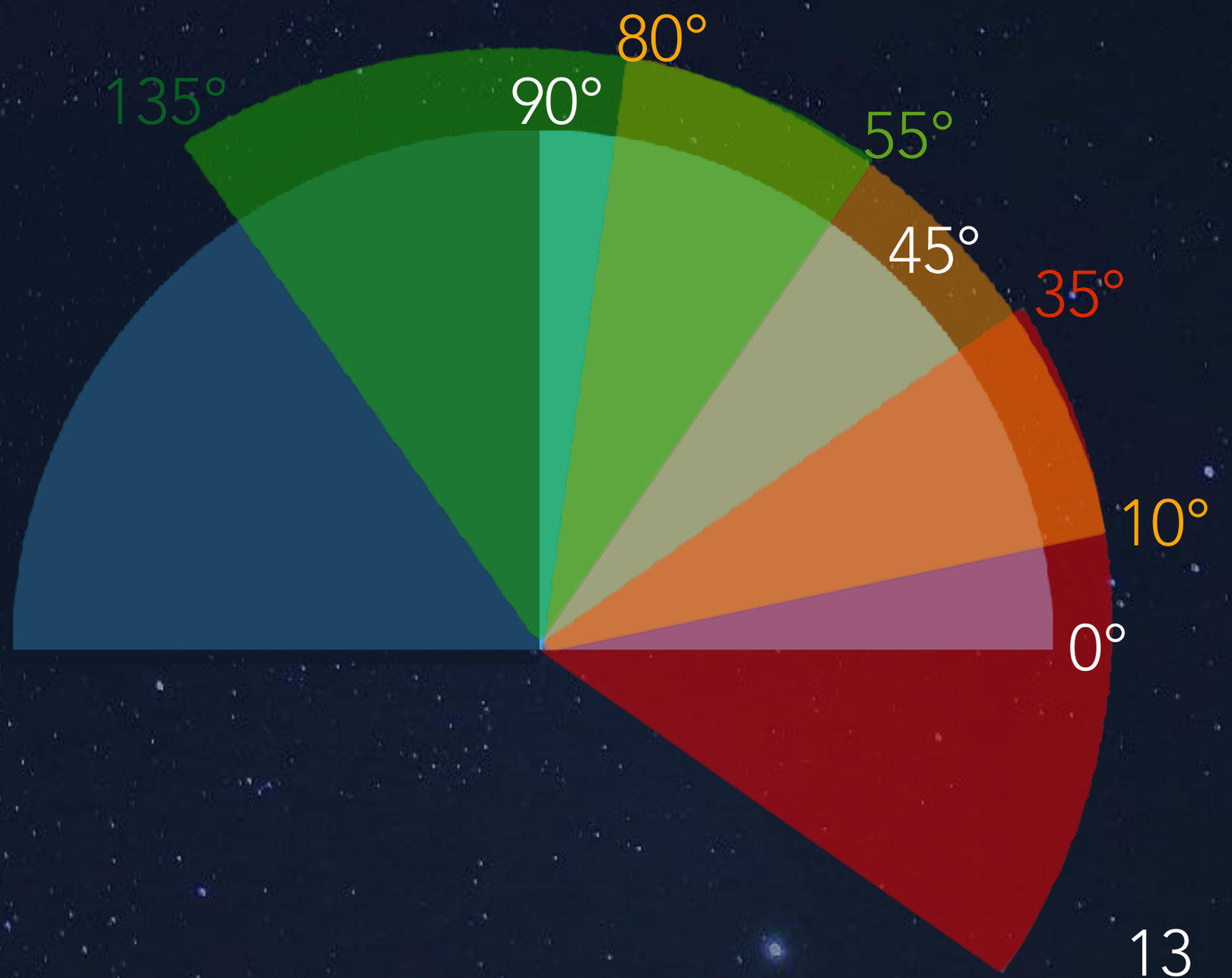
→ solution : on segmente en 3 photos respectivement réalisées avec des angles de 0° (horizon), 45°, 90° (azimut).



LIMITES DE CE PROTOCOLE

1) La plupart des smartphones ne disposent pas d'une résolution angulaire suffisante.

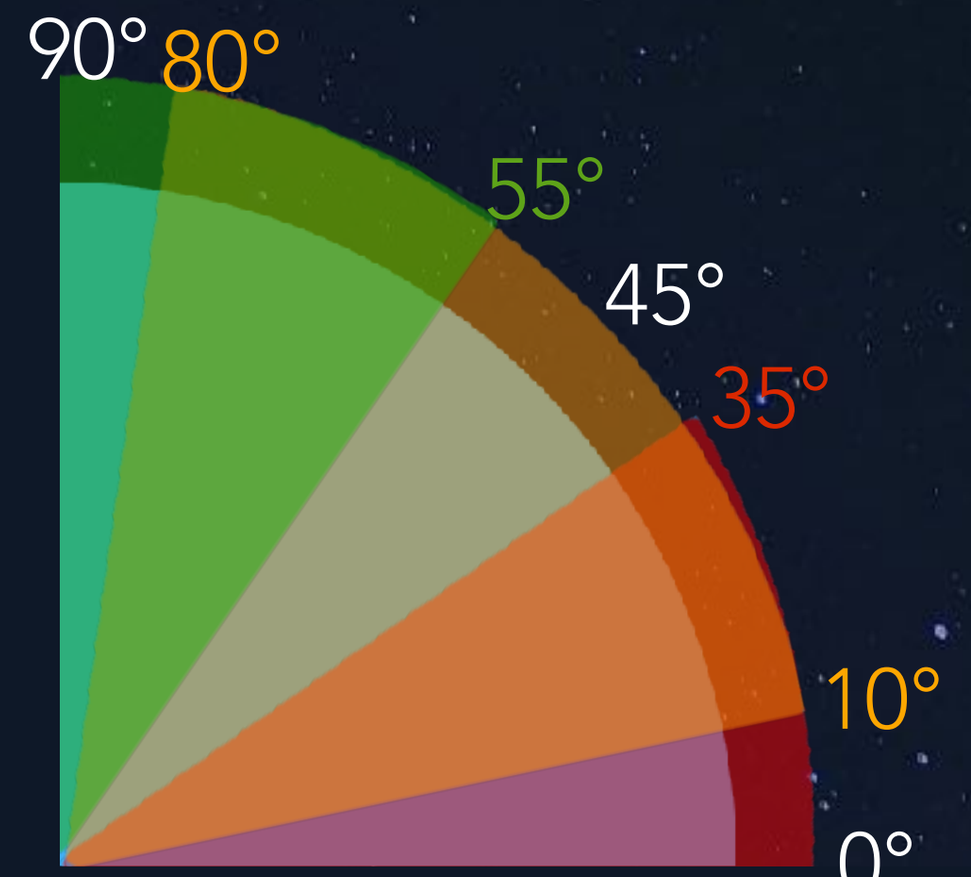
→ solution : on segmente en 3 photos respectivement réalisées avec des angles de 0° (horizon), 45°, 90° (azimut).



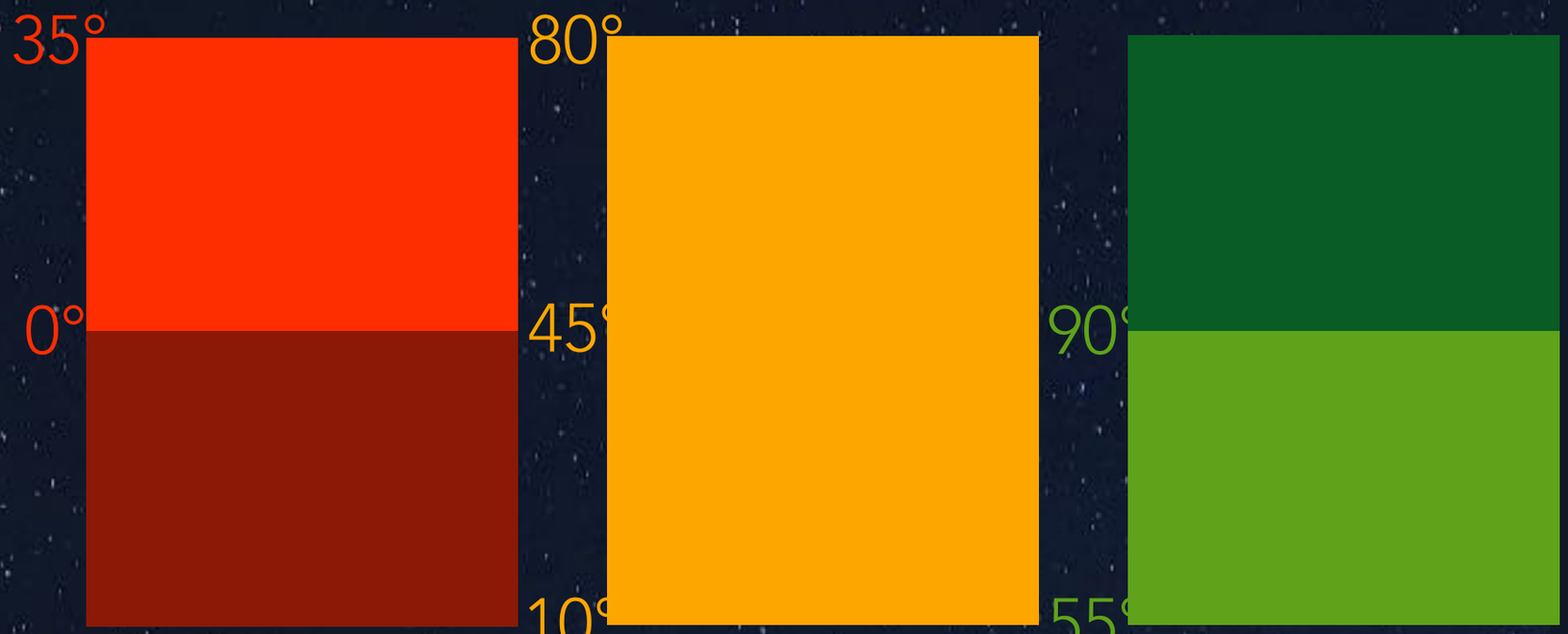
LIMITES DE CE PROTOCOLE

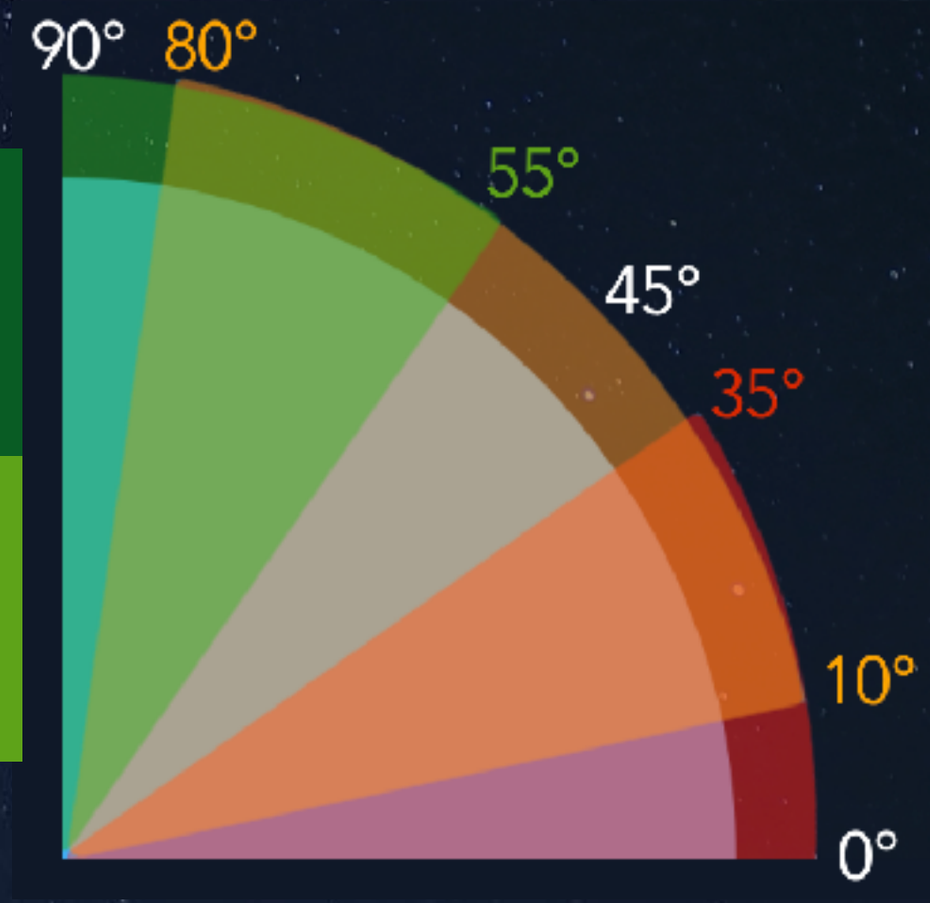
1) La plupart des smartphones ne disposent pas d'une résolution angulaire suffisante.

Conclusion : au final, on couvre bien les angles compris entre 0° et 90°

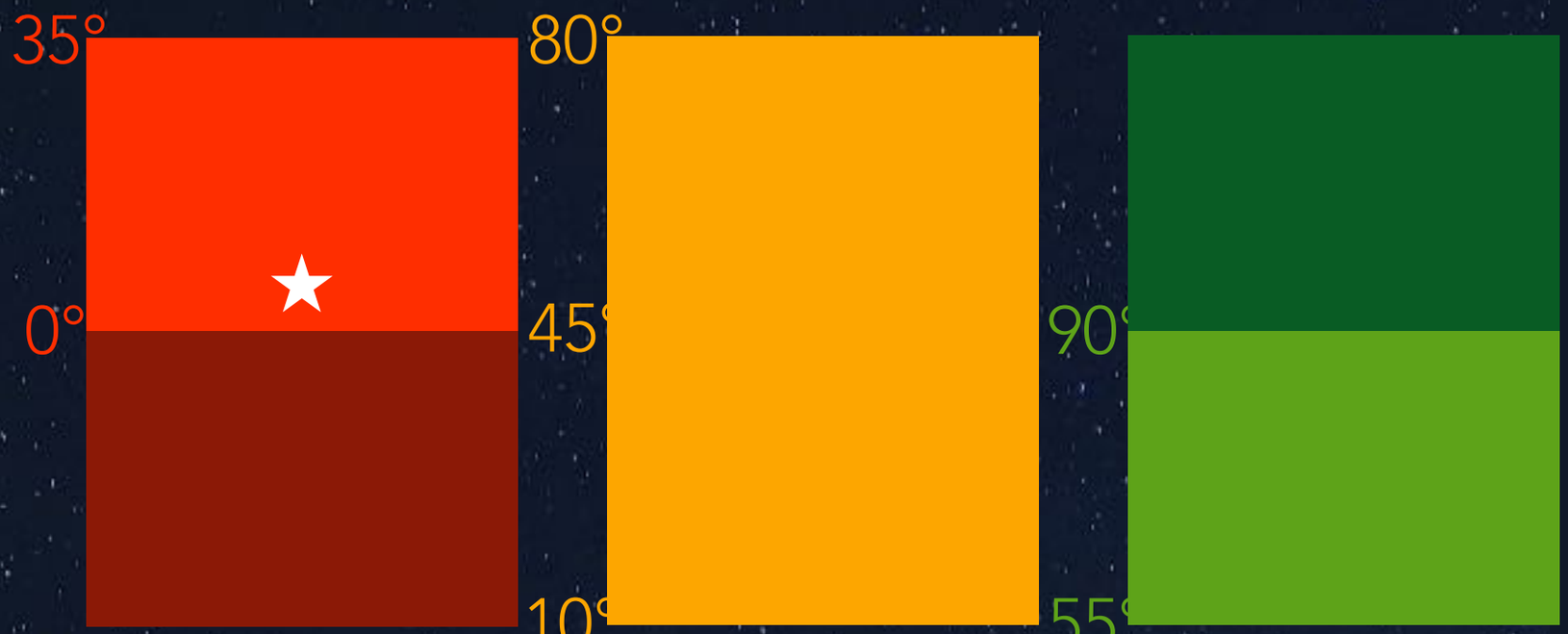


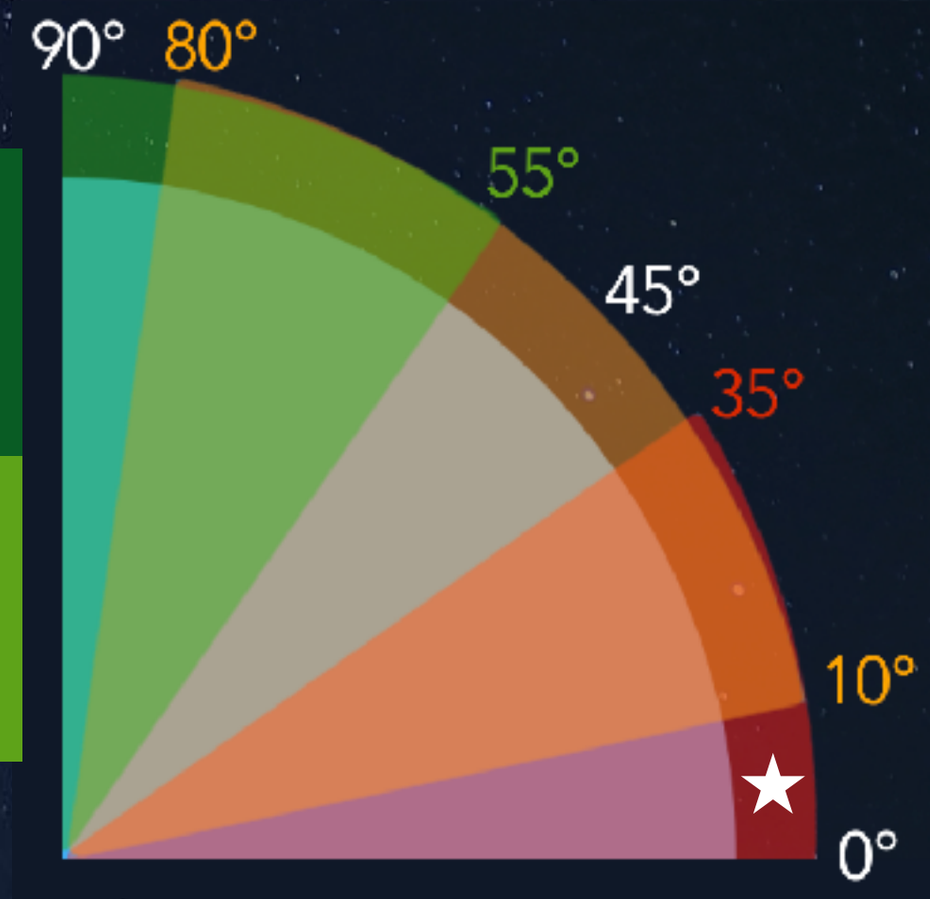
Remarque : on observe différents cas :



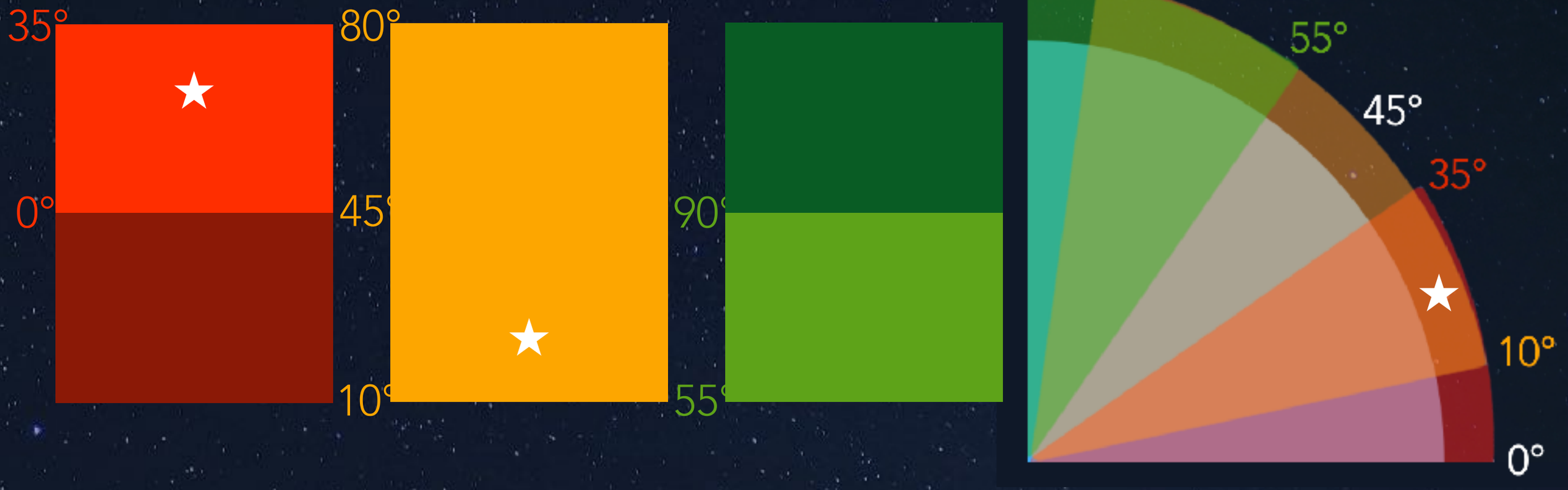


Remarque : on observe différents cas :

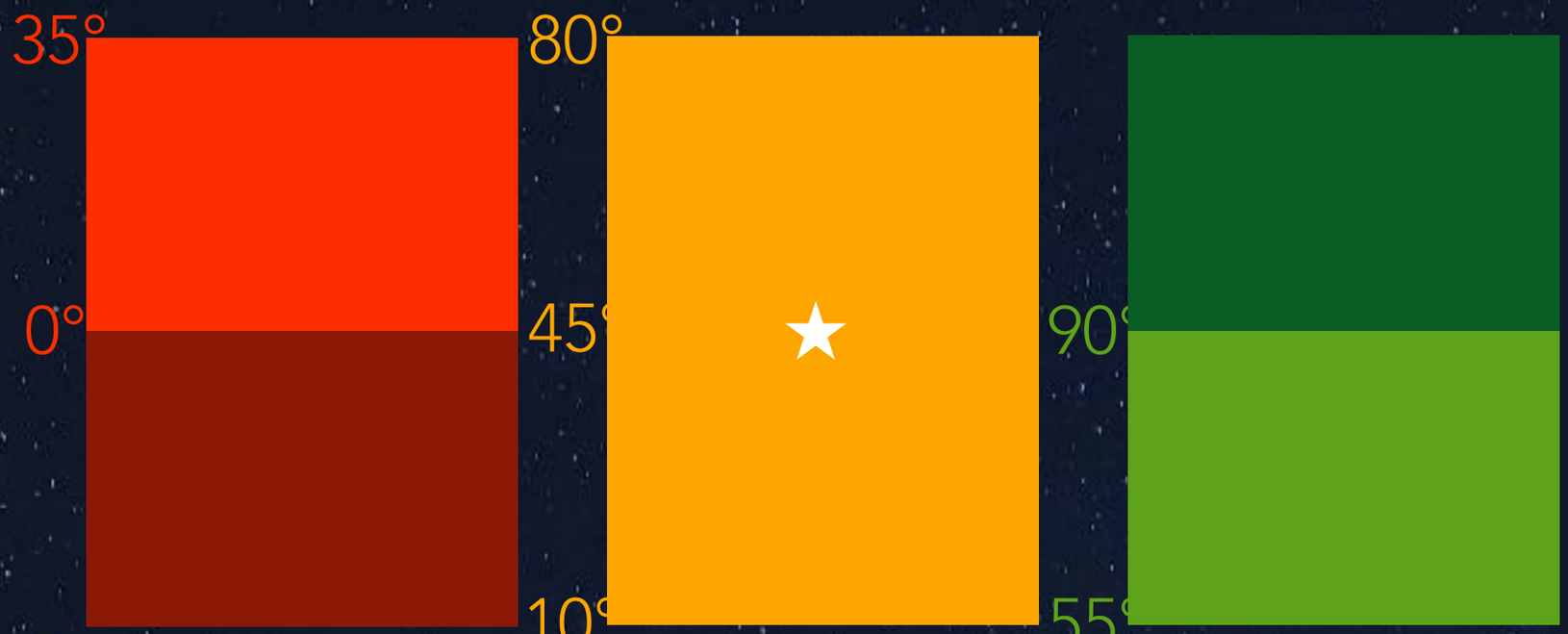


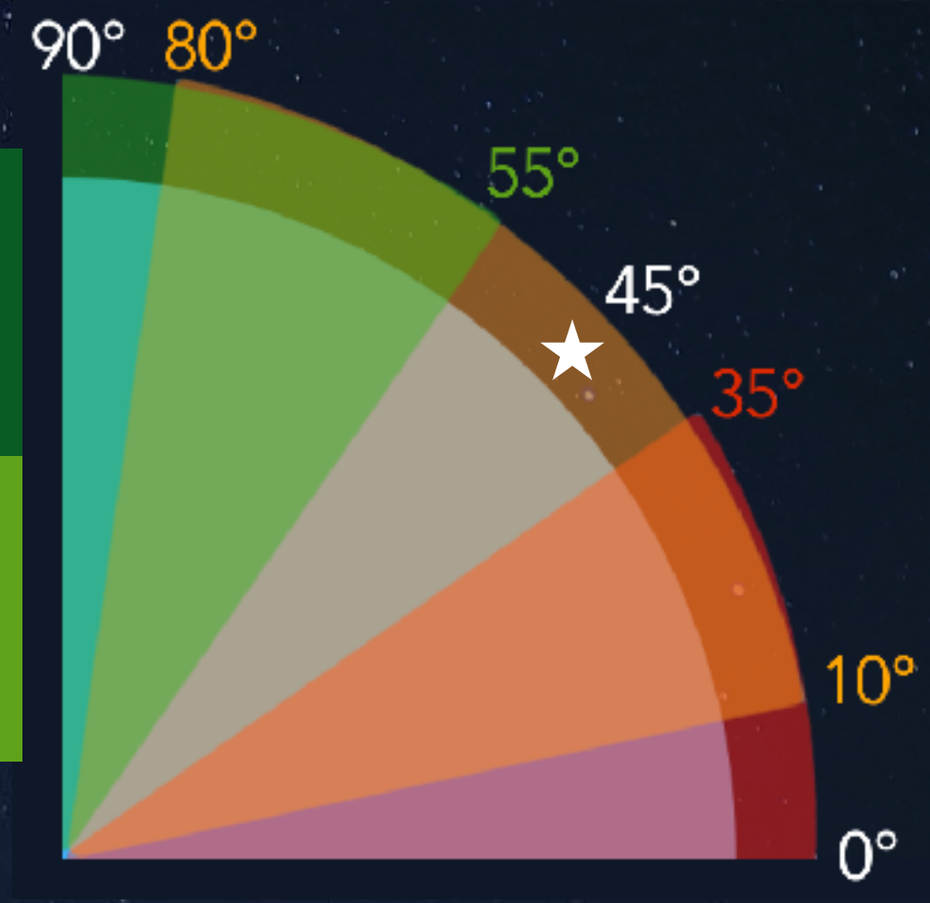


Remarque : on observe différents cas :

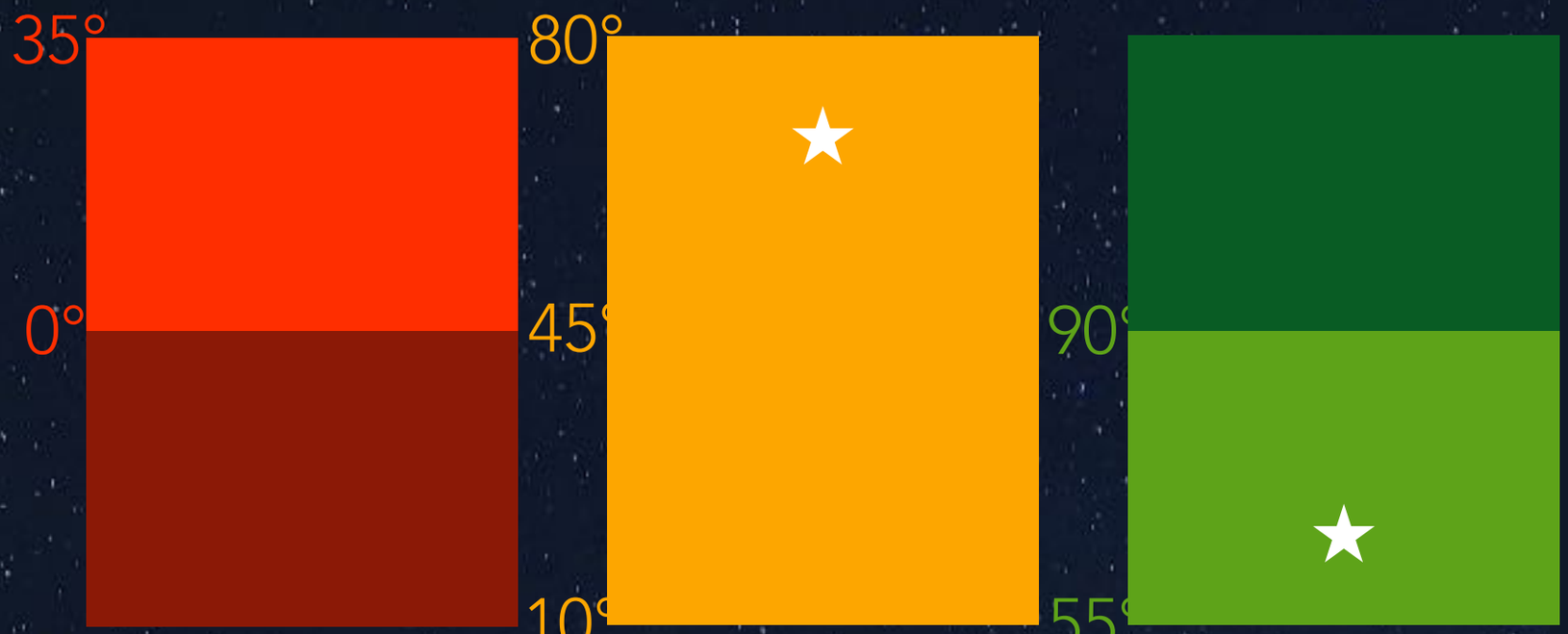


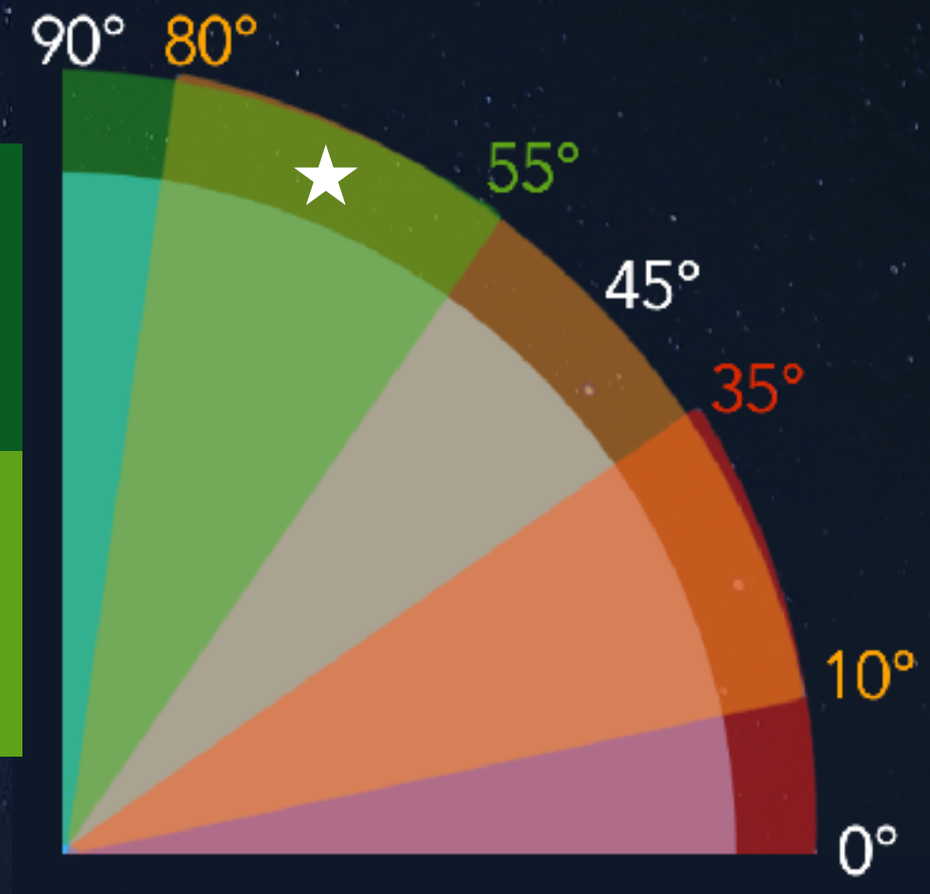
Remarque : on observe différents cas :



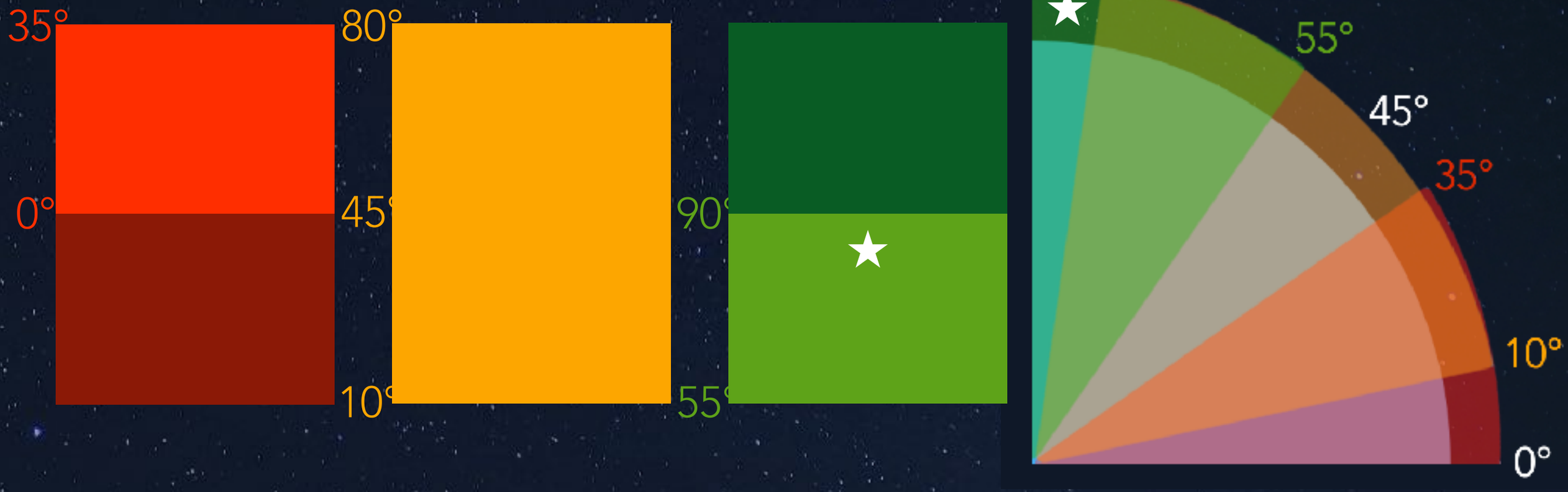


Remarque : on observe différents cas :





Remarque : on observe différents cas :



2) Comment mesurer la hauteur (angle) de l'étoile polaire (qui est égale à la latitude de l'observateur) sur une photo ?

→ **Faire le lien entre la hauteur et la distance mesurée** entre l'étoile polaire et l'horizon : méthode de la régression linéaire (mais puisqu'on dispose de 3 photos, on va en réaliser 3 différentes).

→ Grâce au **logiciel Stellarium**, on effectue cette étude numériquement.

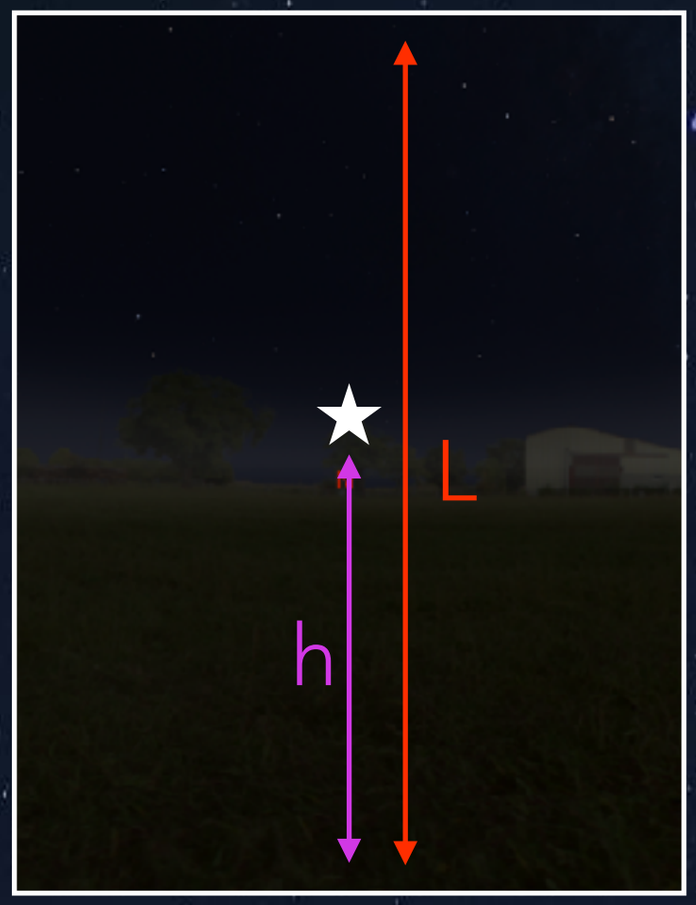
Exemple : régression pour la photo à 0°

Latitude : 5°

Latitude : 10°

Latitude : 20°

Exemple : Régression pour la photo à 0°



Latitude : 5°

Rapport h/L = 0,57

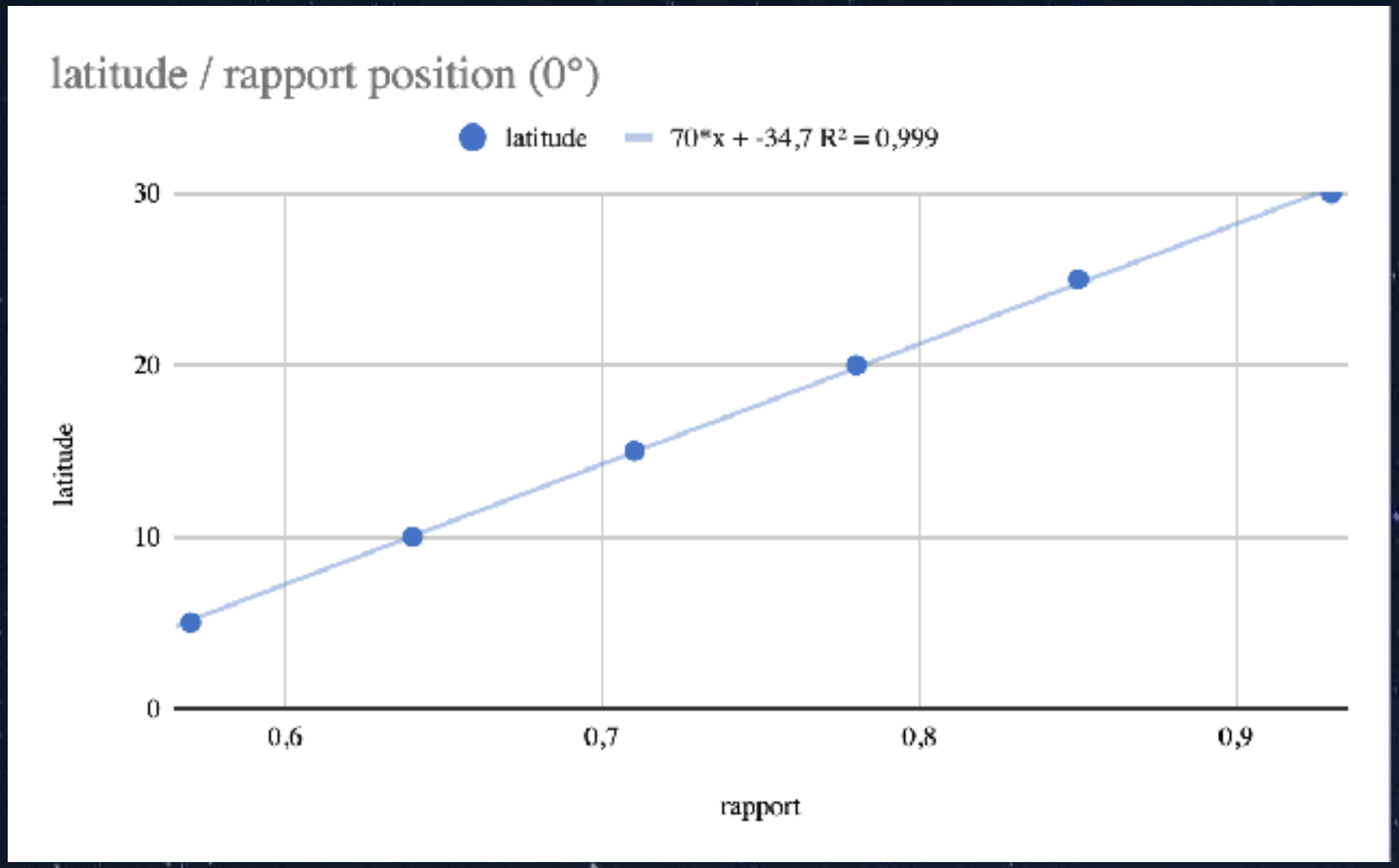
Latitude : 10°

Rapport h/L = 0,65

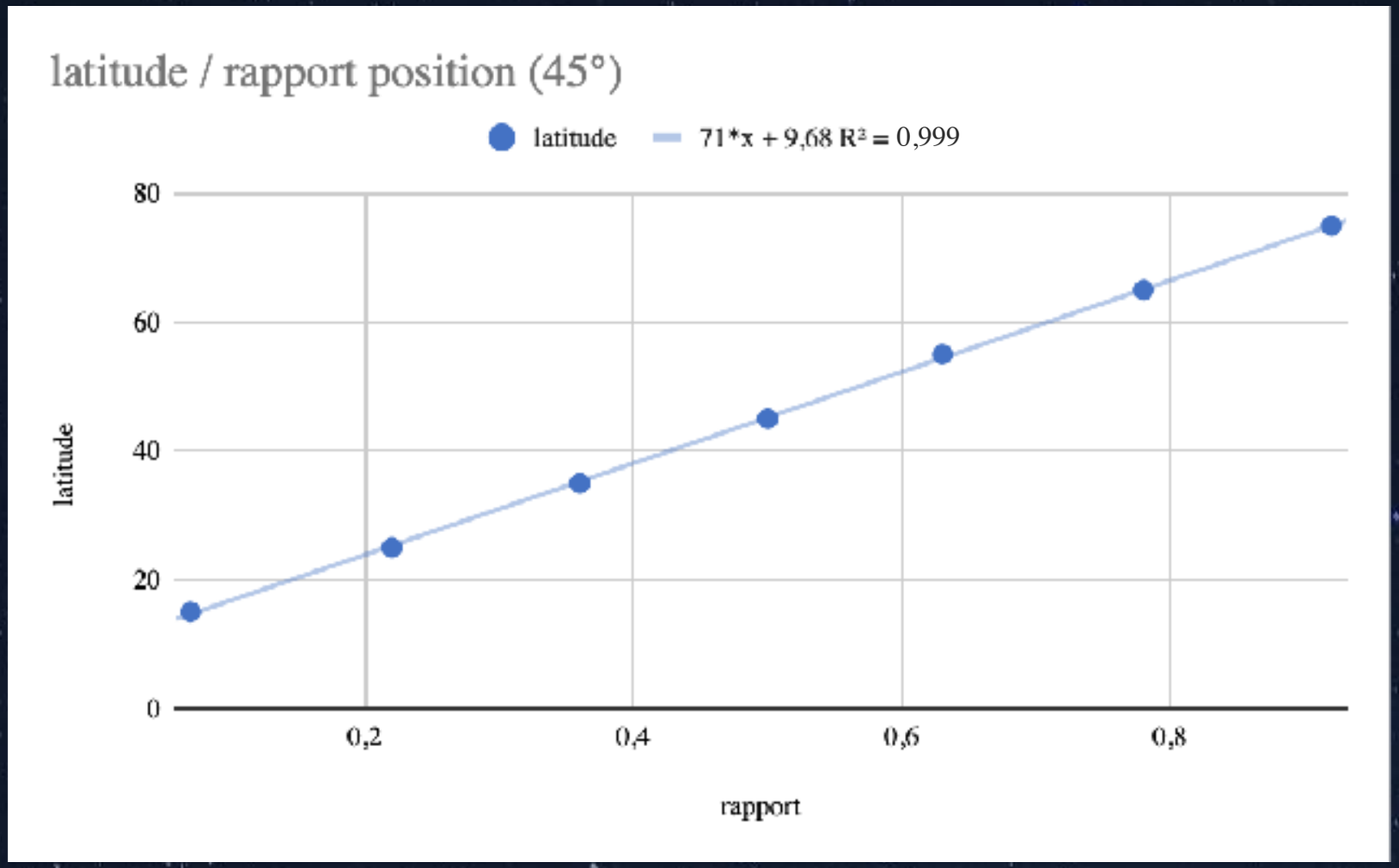
Latitude : 20°

Rapport h/L = 0,78

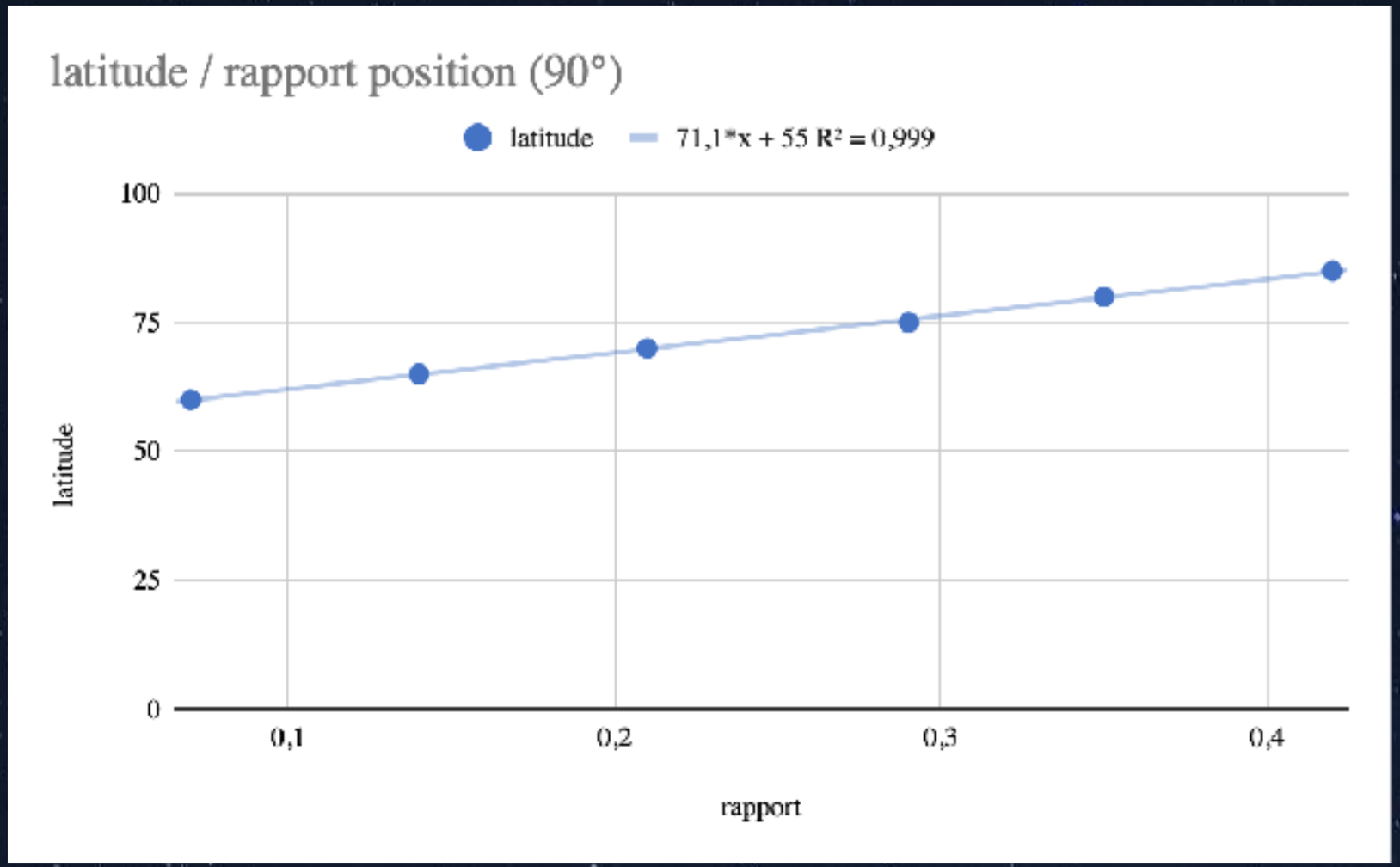
On obtient finalement :



Régression linéaire entre le rapport mesuré et la latitude pour la photo à 0°



Régression linéaire entre le rapport mesuré et la latitude pour la photo à 45°



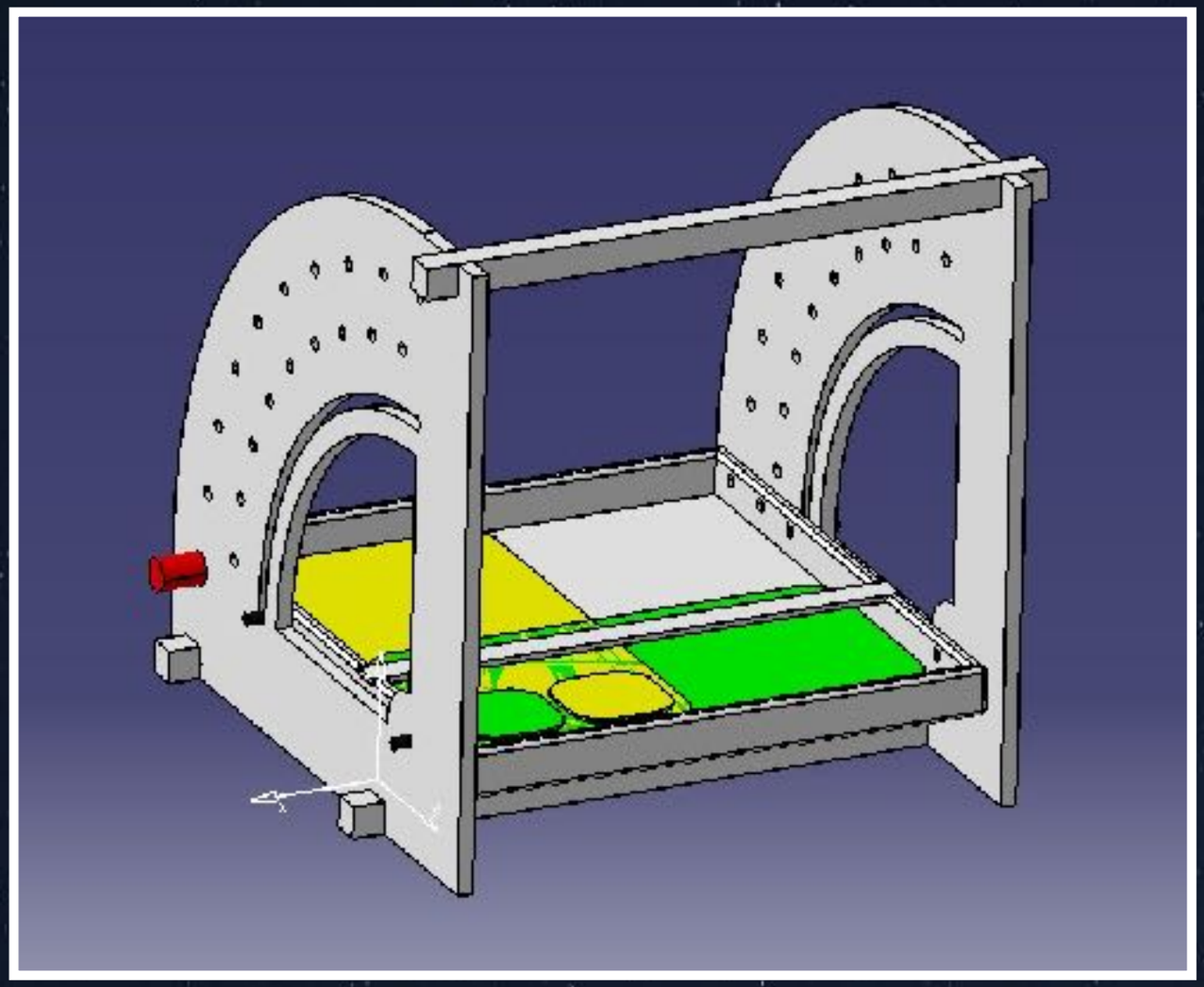
Régression linéaire entre le rapport mesuré et la latitude pour la photo à 90°

DÉTERMINATION DE LA LONGITUDE

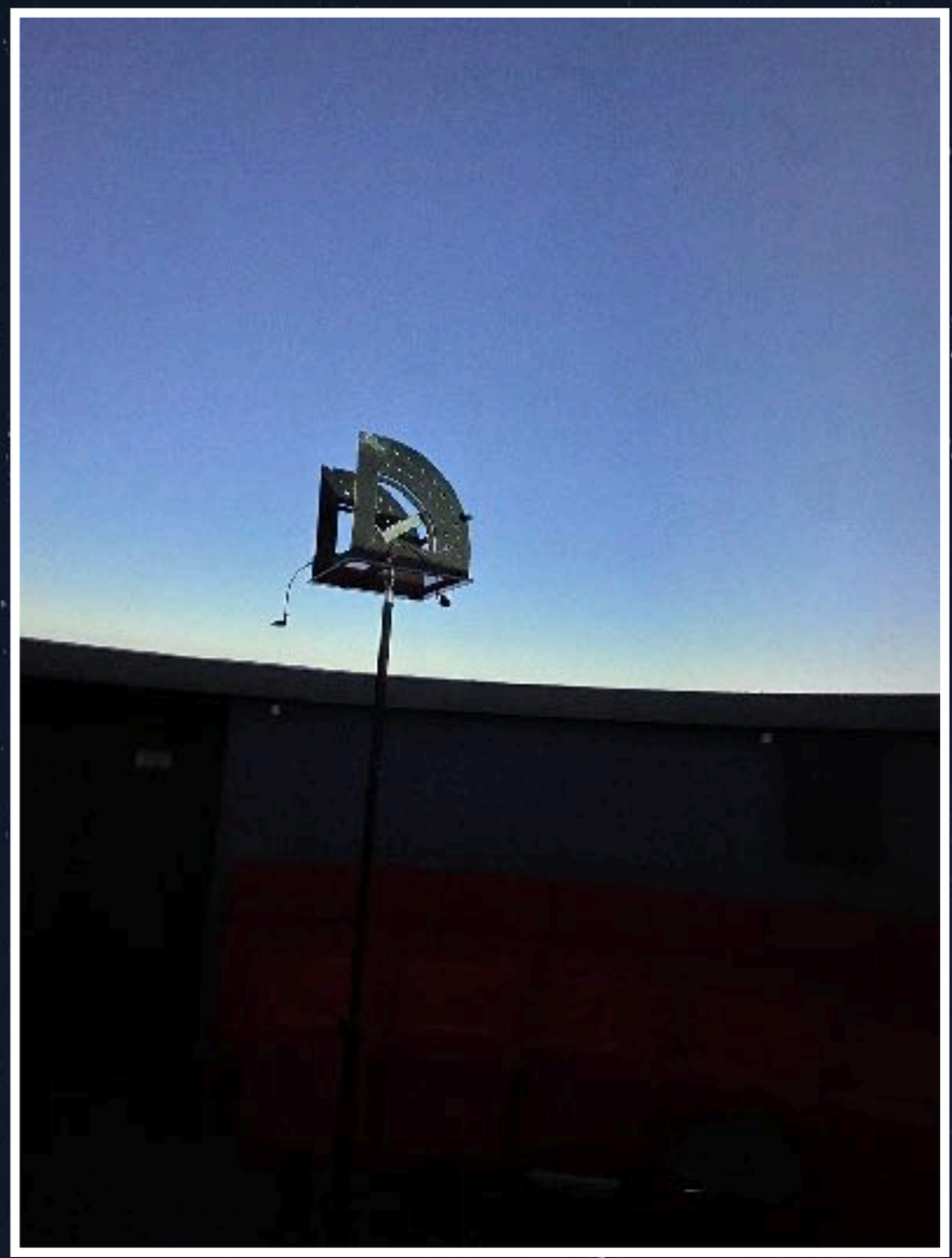
Quelques pistes ?

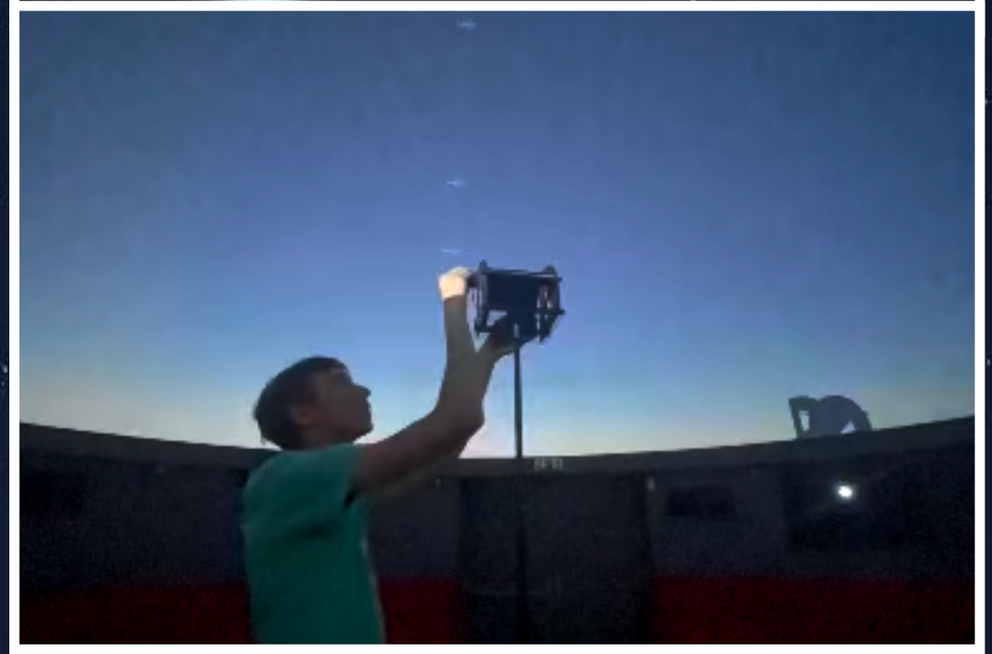
EN PRATIQUE : TROUVER SA POSITION SUR TERRE À PARTIR DE PHOTOGRAPHIES ADAPTÉES

CRÉATION D'UN OUTIL PERMETTANT DE RÉALISER LES PHOTOS NÉCESSAIRES



VISITE DU PLANÉTARIUM DE NANTES



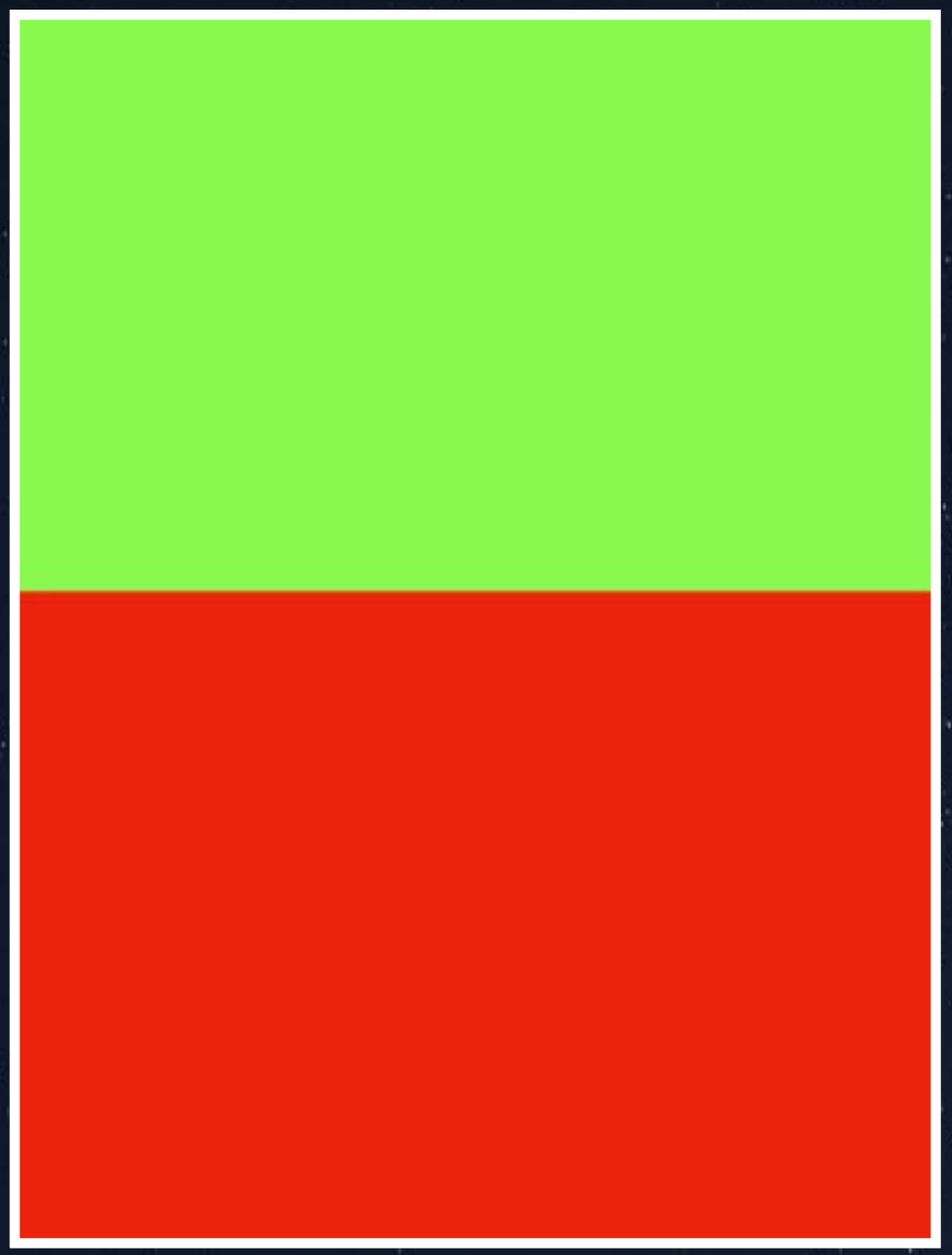


Notre contact : **Daniel Audéon**, médiateur scientifique au planétarium

Application : on teste les images obtenues et on vérifie leur validité par rapport au modèle précédent.

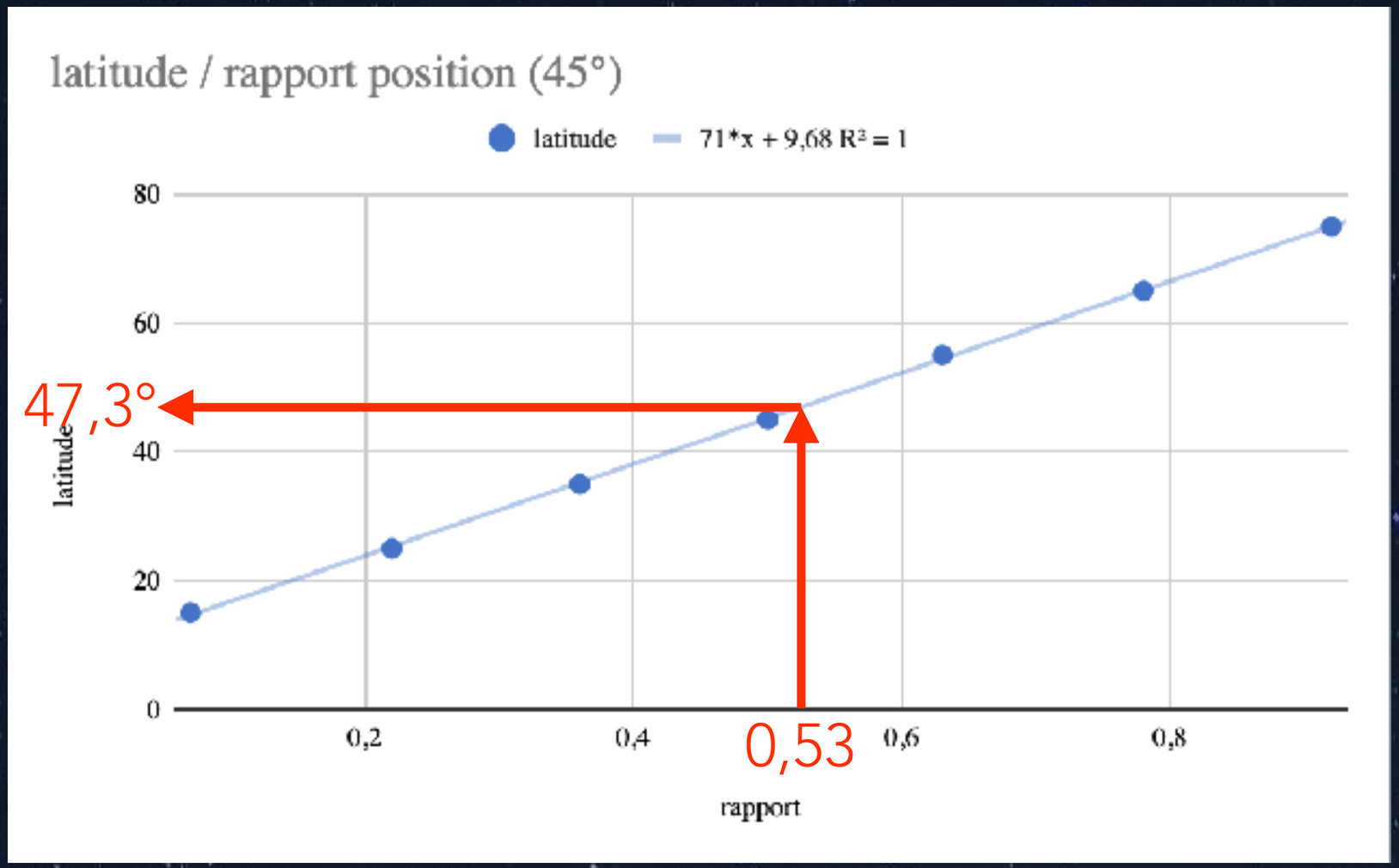
Simulation du Ciel à Nantes
(latitude : 48°N)

Application : on teste les images obtenues et on vérifie leur validité par rapport au modèle précédent.



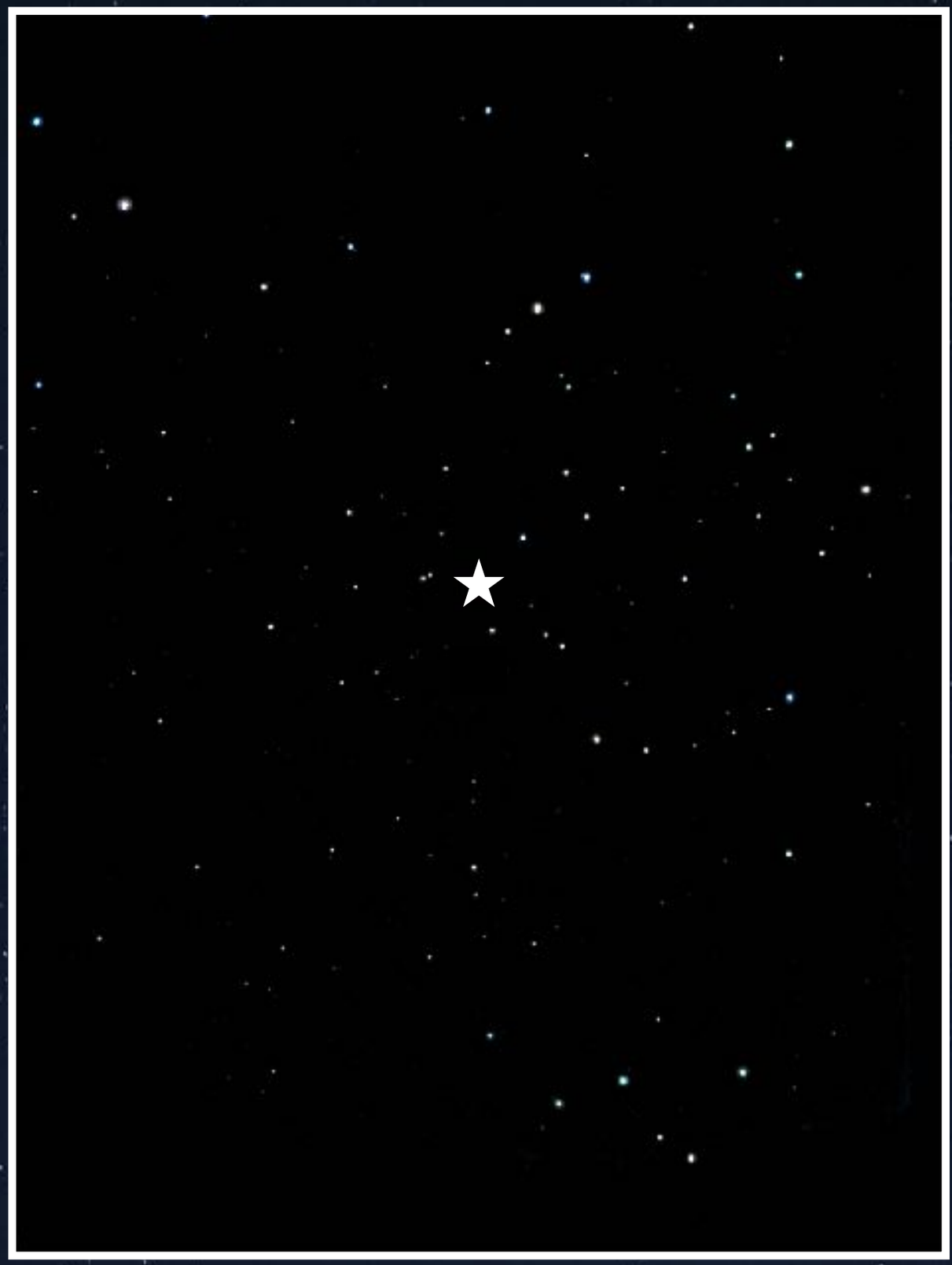
Simulation du Ciel à Nantes
(latitude : 48°N)

Rapport (proportion de rouge): 53 %



Régression linéaire entre le rapport mesuré et la latitude pour la photo à 45°

Exemple réel :



Latitude trouvée :
46,3°N
Latitude réelle :
45,5°N

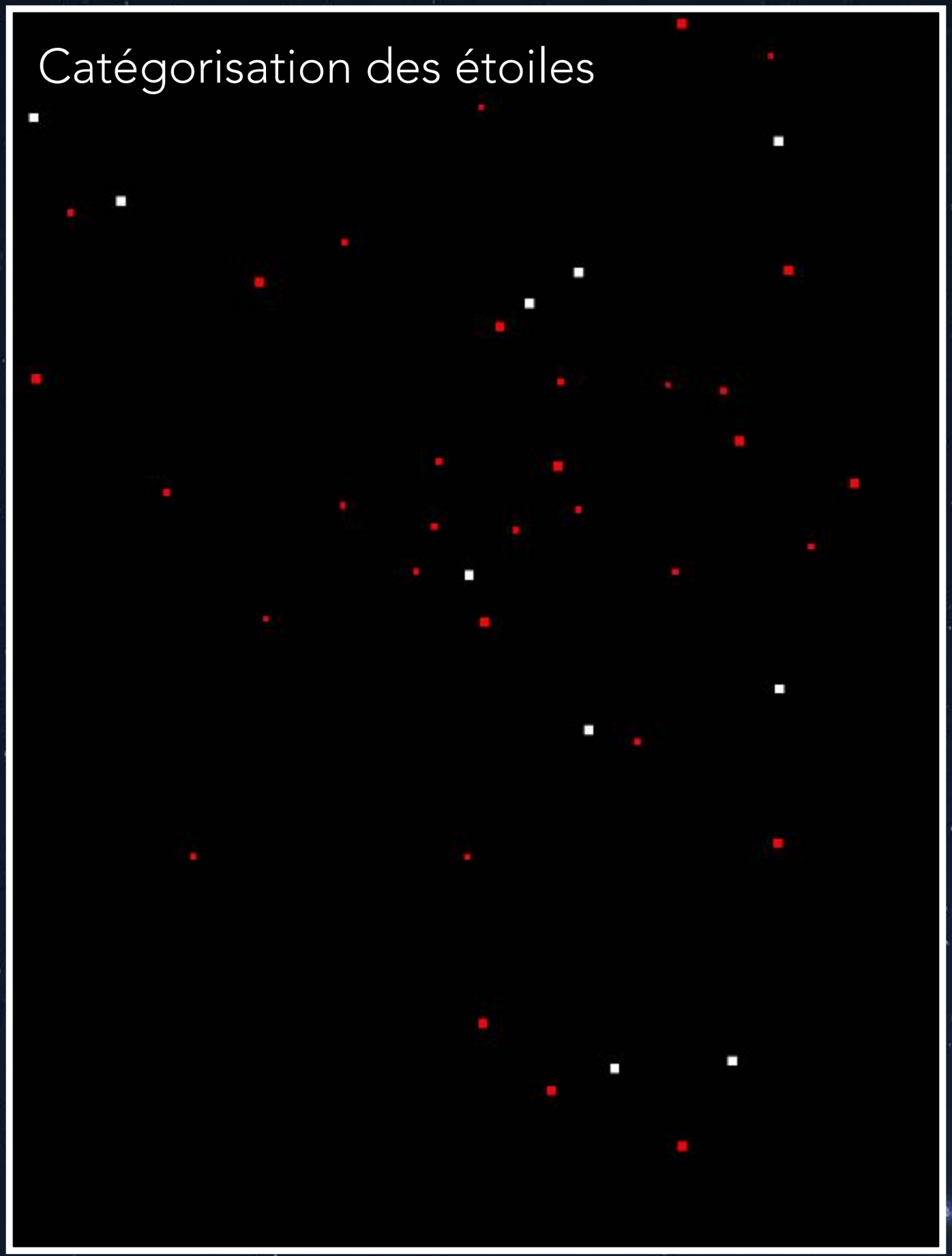
Ciel d'Auvergne : photo
réalisée avec un angle de 45°

VERS UN PROGRAMME INFORMATIQUE

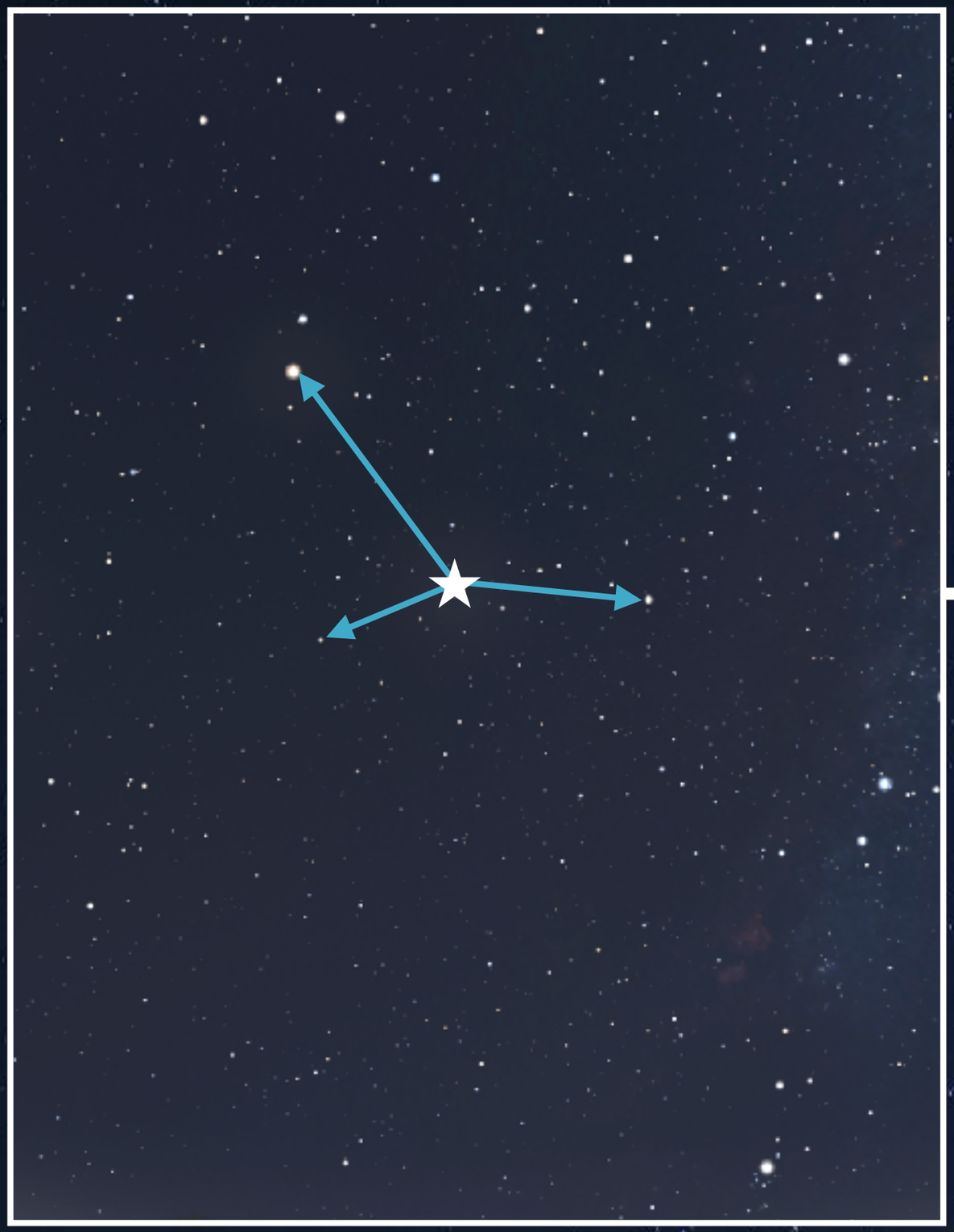
Objectif : créer un **programme** permettant d'**automatiser** la **détection de l'étoile polaire** sur une photo puis d'en déduire la latitude et la longitude de l'observateur.

Image originale

Catégorisation des étoiles



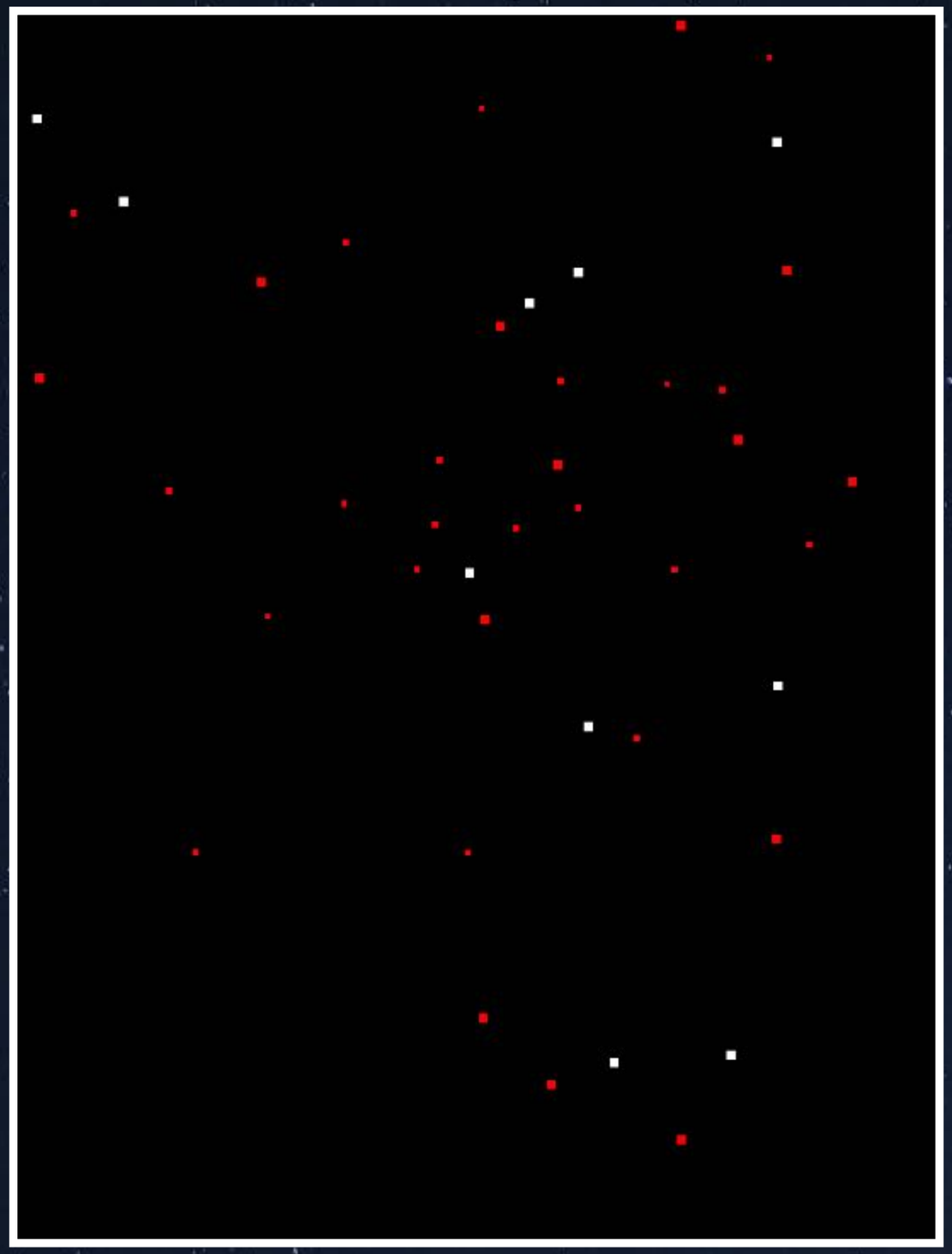
Identification de l’étoile polaire :



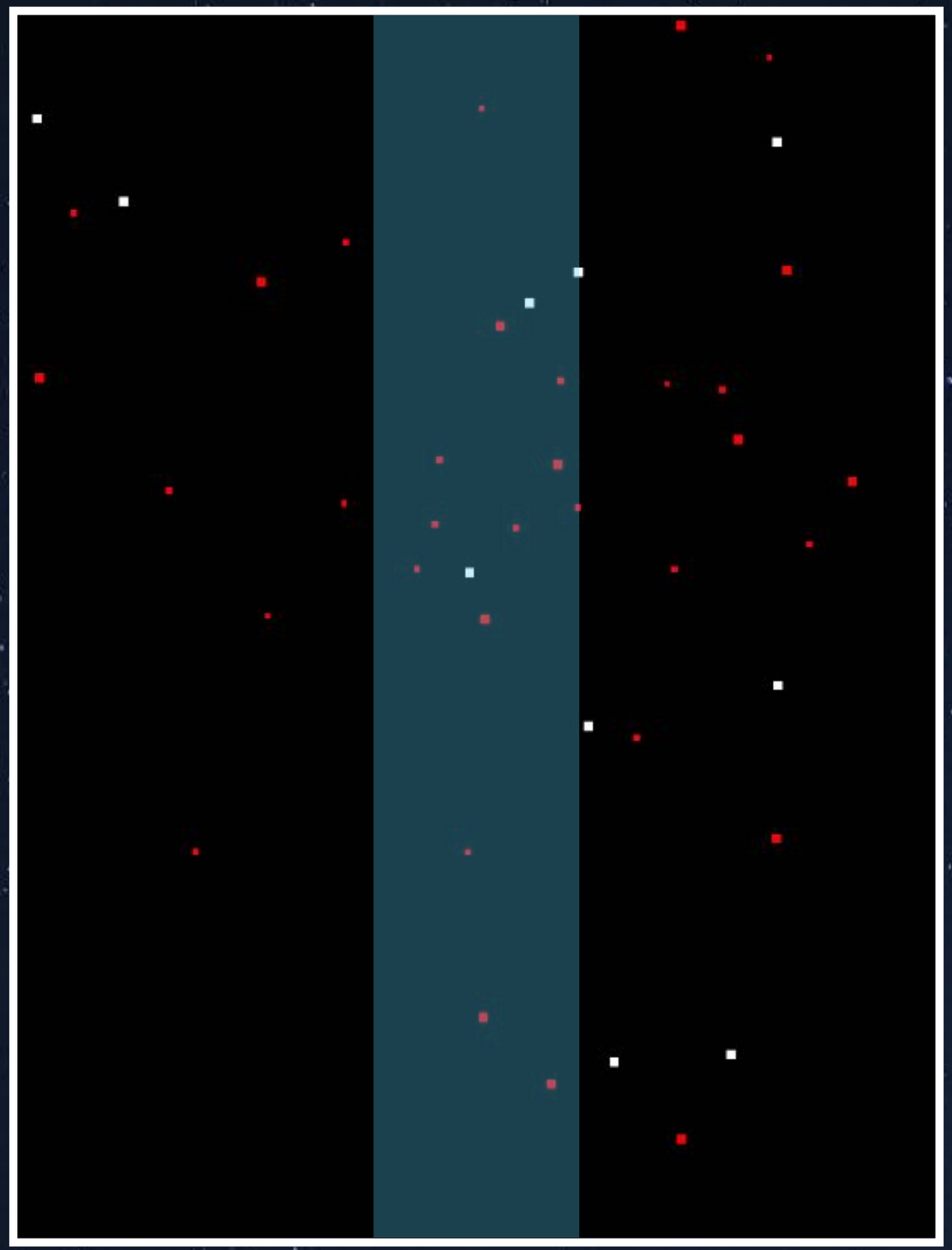
Ciel de référence (Stellarium)

Distances de référence

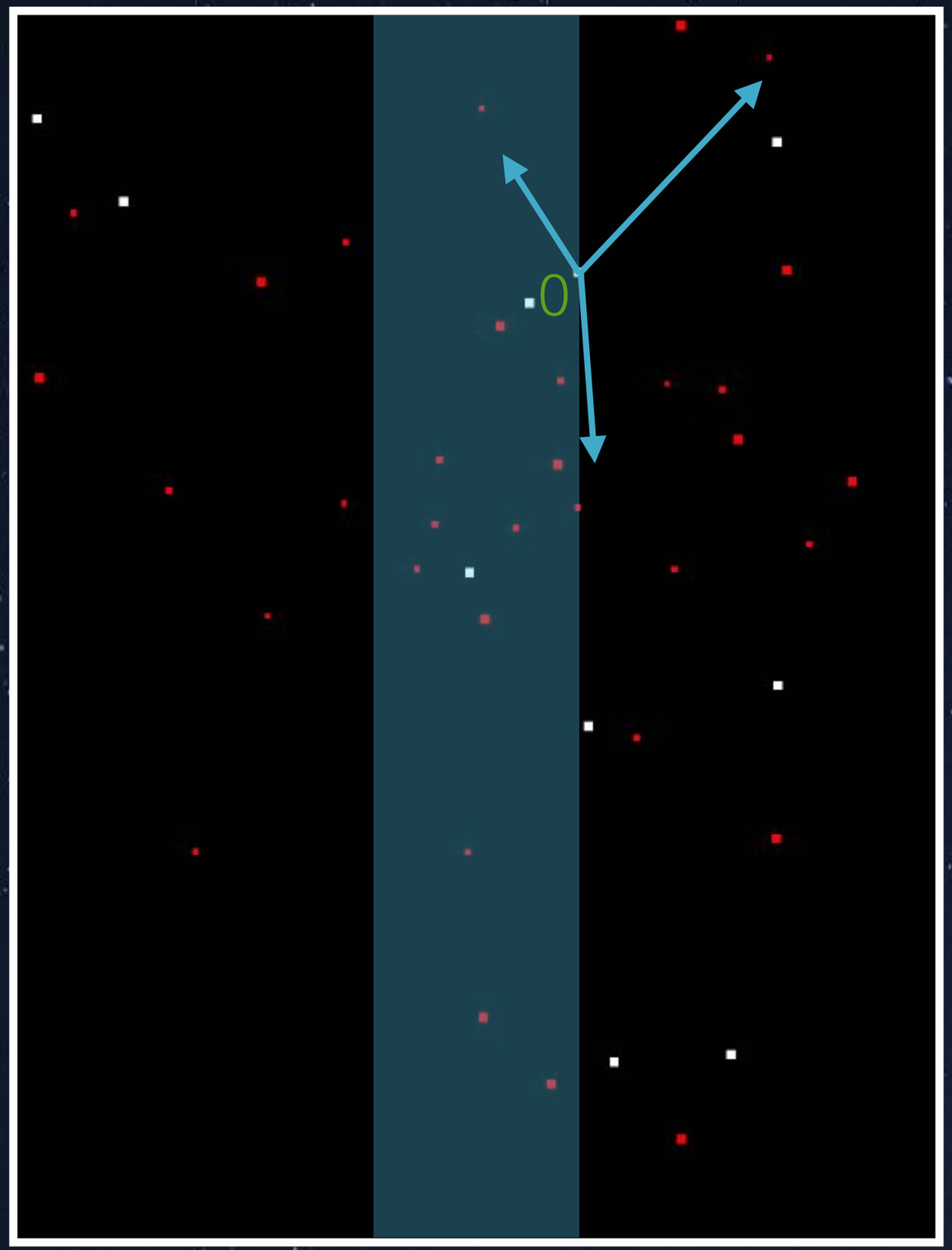
Identification de l’étoile polaire :



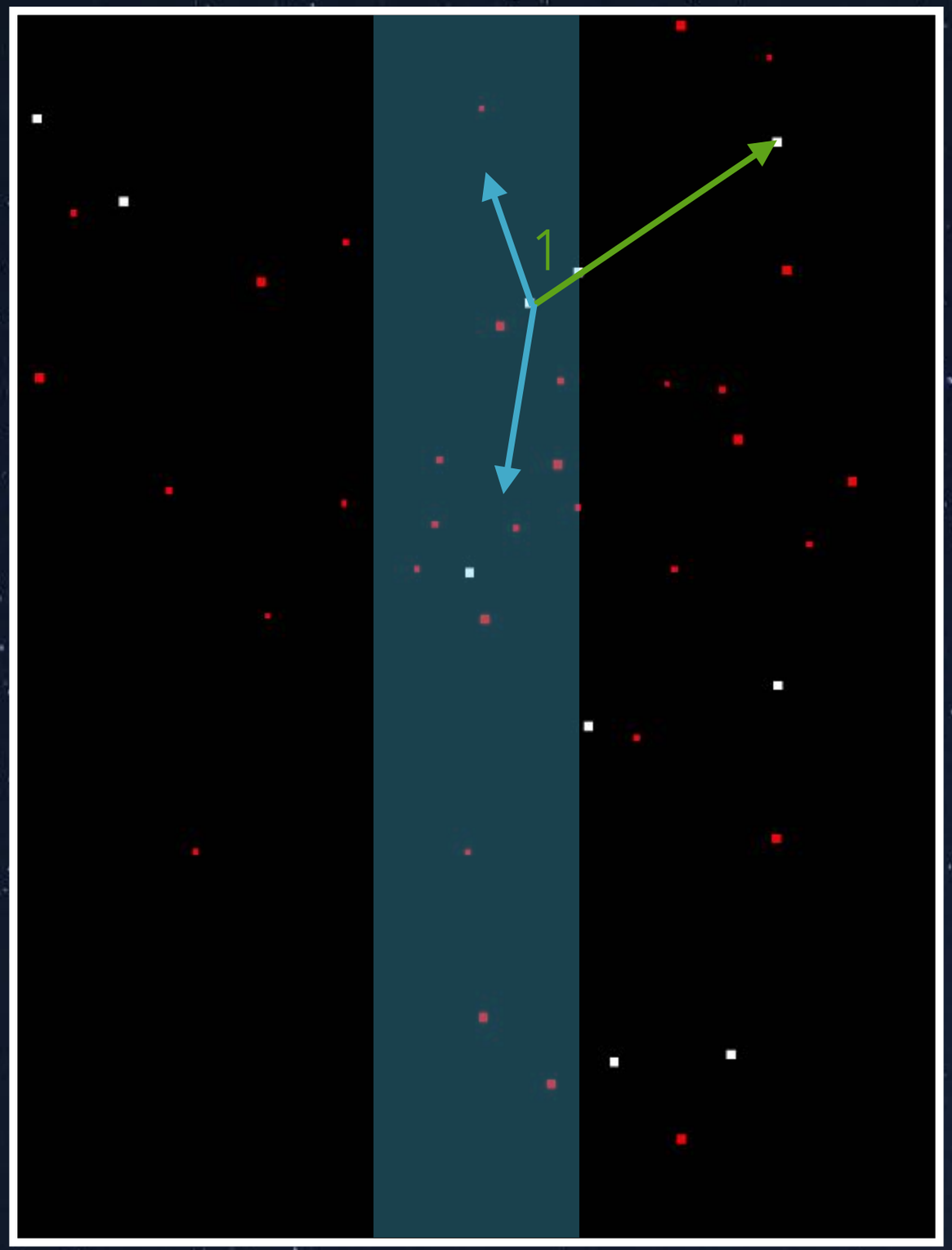
Identification de l’étoile polaire :



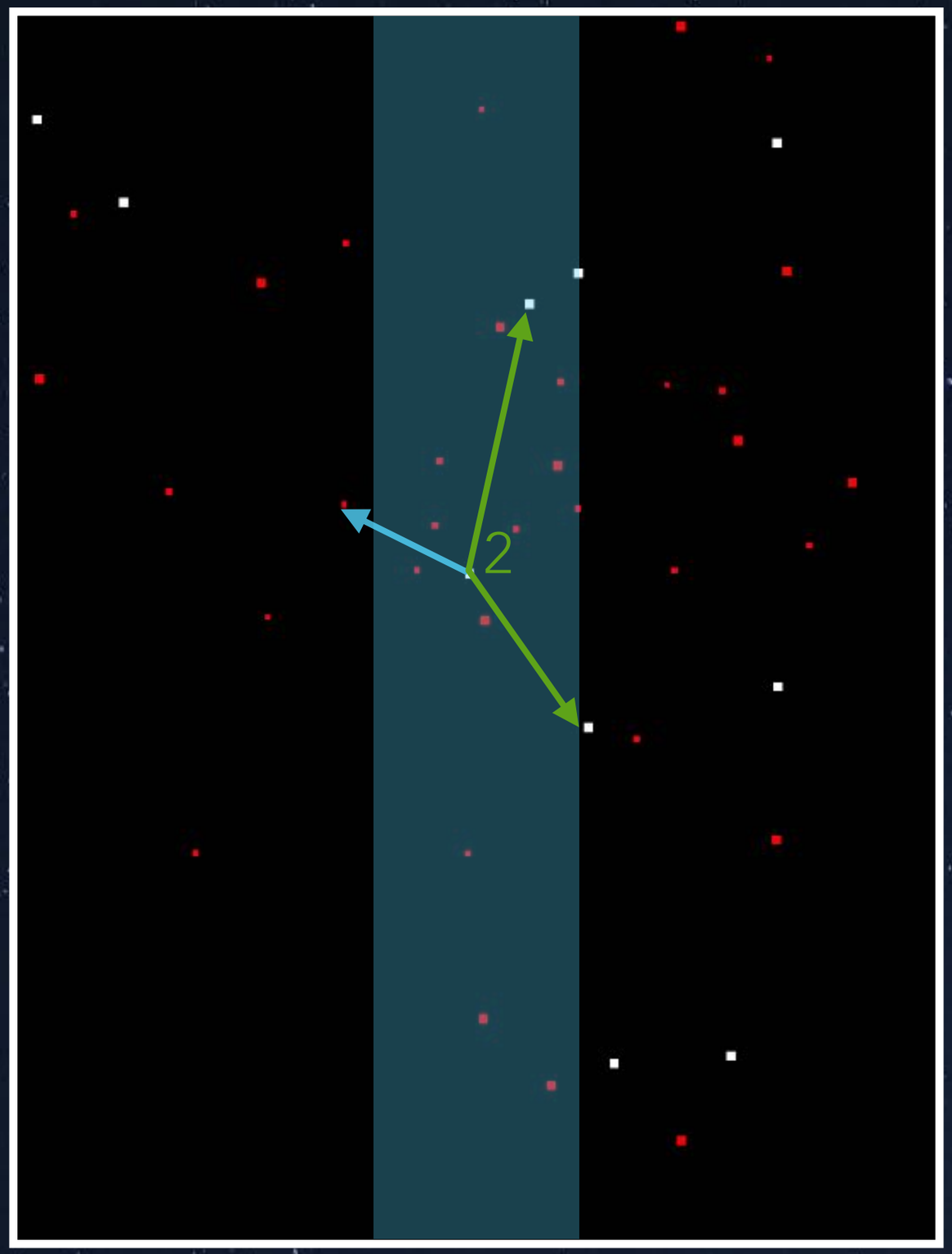
Identification de l’étoile polaire :



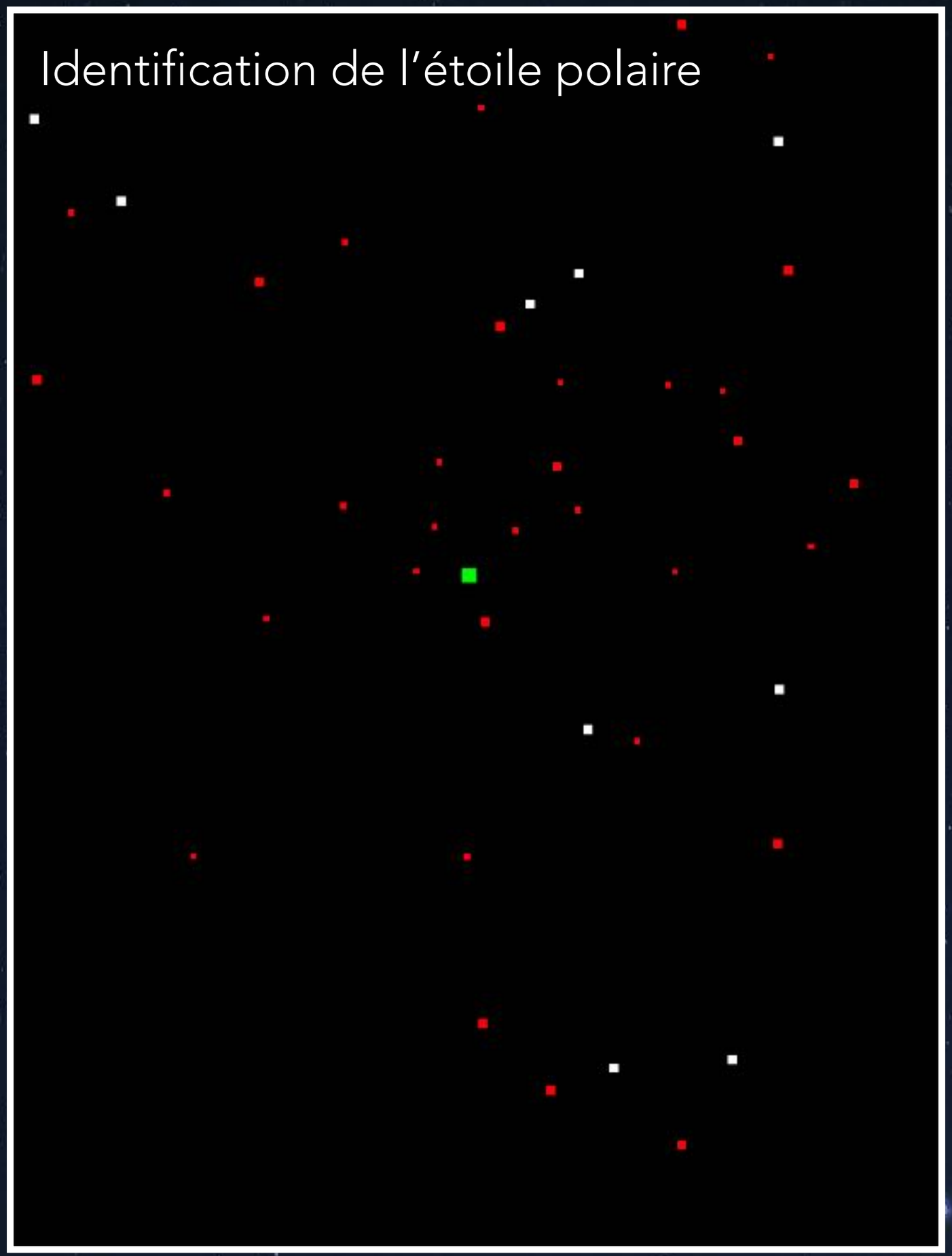
Identification de l’étoile polaire :



Identification de l’étoile polaire :



Identification de l’étoile polaire



```
>>> %Run TIPE3.py

  (3024, 4032)
  [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
  []
  [((92, 629), 94, 'moyenne'), ((132, 2198), 122, 'grosse'), ((235, 2490 …
  M2 =  [(((404, 1541), 106, 'grosse'), [711.0787579445754, 0.0, 1462.418886639529, 980.1918
  61, 1377.1601214092718, 1732.4168089694813, 1377.8987626092128, 1793.8032222069398, 1531.4
  3072677, 3224.142831823677, 3456.21367973683031]), (((1935, 1503), 264, 'grosse'), [1932.31
  9852176, 1053.6816407245597, 1181.077474173477, 911.7587400184327, 985.6596775763935, 1295
  .720822892431, 1803.758853062127, 1678.6080543116668, 1706.0228603392159, 1987.827960362767
  etoile =  (((1935, 1503), 264, 'grosse'), 7)
  latitude =  46.606339285714284
```

Latitude renvoyée par le programme

Travail de mon binôme :

Reconnaissance de l'étoile polaire grâce à un réseau de neurones (Intelligence artificielle et principe du Deep-Learning)

Source Image : https://youtu.be/aircAruvnKk?si=bmhjH0X55xcBlxVW

MÉTHODE	COMPARAISON AUX DISTANCES DE RÉFÉRENCE	RÉSEAU DE NEURONES
AVANTAGES	Fonctionnement testé et efficace Complexité temporelle moindre Plus intuitif	Peut fonctionner sur des cas complexes Plus précis
INCONVÉNIENTS	Nécessite d'adapter les paramètres du programme Plus faillible	Nécessite une large base de données Actuellement non fonctionnel avec des cas réels

CONCLUSION

Source Image : https://blog.loisirsplaisirs.com/stage-astronomie/

ANNEXE

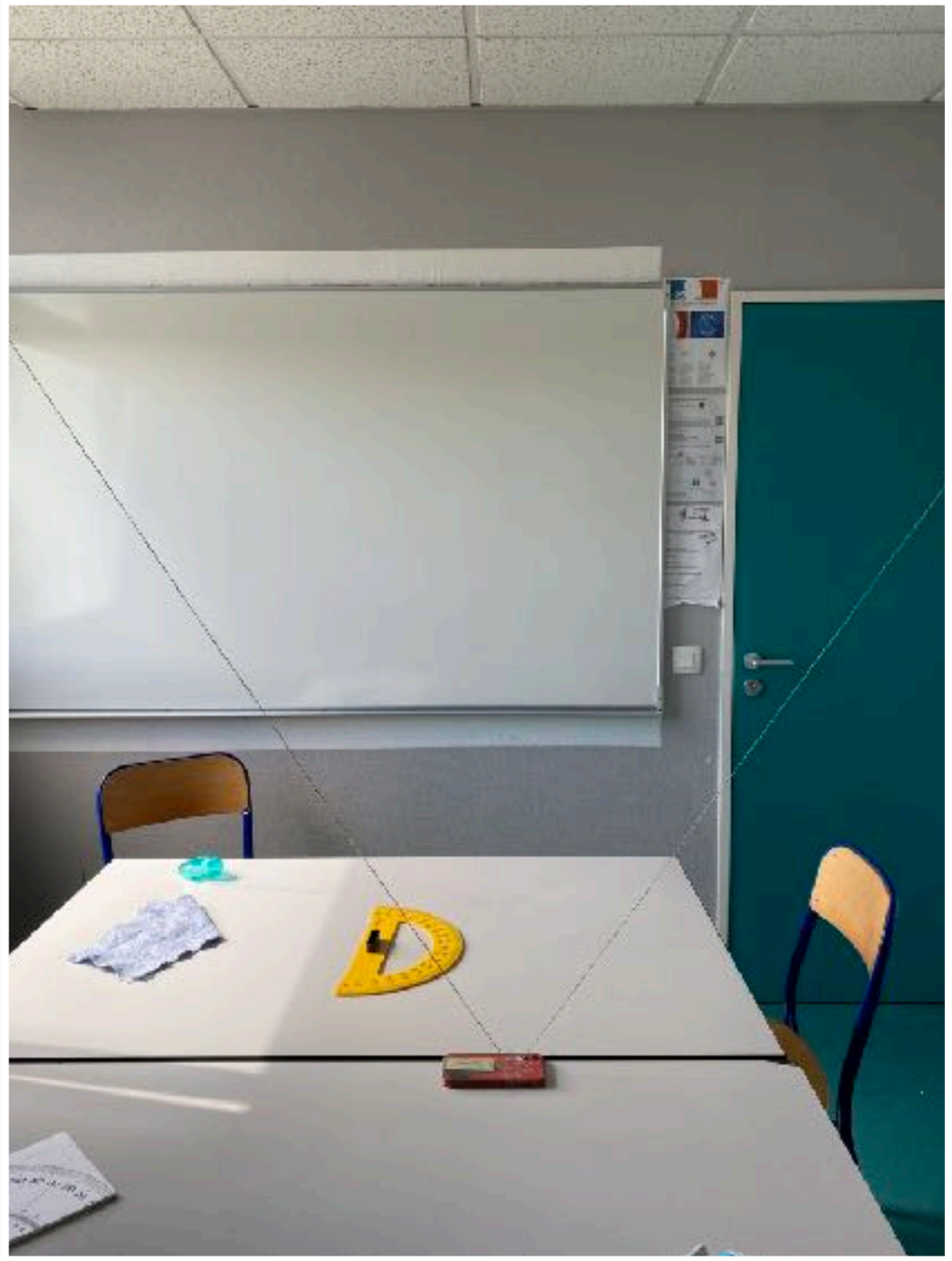
```
photo=Image.open("auvergne2.jpg")
seuil = 100
lst =[]
print(photo.size)

def moyenne_points(lst):
    L = len(lst)
    x=0
    y=0
    for i in range(L):
        x = x + lst[i][0]
        y = y + lst[i][1]
    return(int((x/L)),int((y/L)))

def matrice(photo,seuil):
    cols,rows = photo.size
    matrice = []
    for a in range(rows):
        matrice.append([0 for b in range(cols)])
    for i in range(rows):
        for j in range(cols):
            r,v,b = photo.getpixel((j,i))
            if r >= seuil and v >= seuil and b >= seuil :
                matrice[i][j] = 1
    return matrice


photo_matrice = matrice(photo,seuil)

n = 25 #entier impair

def cara_etoiles(photo_matrice,n):
    m = photo_matrice
    h = len(photo_matrice)
    l = len(photo_matrice[0])
    L = []
    for i in range(n+1,h-n-1):
        for j in range(n+1,l-n-1):
            if m[i][j] == 1 :
                s=1
                lst_pos = []
                for i2 in range(n):
                    for j2 in range(int((n+1)/2)):
                        if m[i+i2][j+j2] == 1 :
                            s+=1
                            lst_pos.append([i+i2,j+j2])
                            m[i+i2][j+j2] = 0
                    for j2 in range(int((n-1)/2)):
                        if m[i+i2][j-j2-1] == 1 :
                            s+=1
                            lst_pos.append([i+i2,j-j2-1])
                            m[i+i2][j-j2-1] = 0
                moyenne = moyenne_points(lst_pos)
                L.append((moyenne,s))
    return L

L= cara_etoiles(photo_matrice,n)
```

```
def image(photo,L_cara):
    cols,rows = photo.size
    black = (0,0,0)
    white = (255,255,255)
    red = (255,0,0)
    yellow = (255,255,0)
    orange = (255,125,0)
    green = (0,255,0)
    blue = (0,255,255)
    im = Image.new('RGBA',(cols+30,rows+30),black)
    for i in range(len(L_cara)):
        if L_cara[i][2] == "petite" :
            x = L_cara[i][0][0]
            y = L_cara[i][0][1]
            for a in range(20):
                for b in range(20):
                    im.putpixel((y-10+a,x-10+b),red)
        if L_cara[i][2] == "moyenne" :
            x = L_cara[i][0][0]
            y = L_cara[i][0][1]
            for a in range(30):
                for b in range(30):
                    im.putpixel((y-15+a,x-15+b),red)
        if L_cara[i][2] == "grosse" :
            x = L_cara[i][0][0]
            y = L_cara[i][0][1]
            for a in range(30):
                for b in range(30):
                    im.putpixel((y-15+a,x-15+b),white)
    return im

def image_etoilesprincipales(photo,L_cara):
    cols,rows = photo.size
    black = (0,0,0)
    white = (255,255,255)
    red = (255,0,0)
    yellow = (255,255,0)
    green = (0,255,255)
    im = Image.new('RGBA',(cols+30,rows+30),black)
    for i in range(len(L_cara)):
        if L_cara[i][2] == "grosse" :
            x = L_cara[i][0][0]
            y = L_cara[i][0][1]
            for a in range(20):
                for b in range(20):
                    im.putpixel((y-10+a,x-10+b),white)
    return im

im = image(photo,L_cara)
im2 = image_etoilesprincipales(photo,L_cara)

print(matrice(photo,seuil))
print(cara_etoiles(photo_matrice,n))
print(categories_etoiles(L))

im.show()
im2.show()
```

```
x1 = (0,0)
x2 = (0,0)

def grosses_etoiles(L_cara):
    lst=[]
    for i in range(len(L_cara)):
        if L_cara[i][2]=="grosse":
            lst.append(L_cara[i])
    return lst

lge = grosses_etoiles(L_cara)

def etoiles_centre(lge,photo):
    cols,rows = photo.size
    lst = []
    milieu = int(cols/2)
    d = int(cols/20)
    for i in range(len(lge)):
        if milieu - d < lge[i][0][1] < milieu + d :
            lst.append(lge[i])
    return lst

lgec = etoiles_centre(lge,photo)

def distance_euclidienne(x1,x2):
    return sqrt((x1[0]-x2[0])**2+(x1[1]-x2[1])**2)

def distance2(lge):
    M =[]
    for i in range(len(lge)):
        M.append((lge[i],[]))
        for j in range(len(lge)):
            M[i][1].append(distance_euclidienne(lge[i][0],lge[j][0]))
    return M

M = distance2(lge)

def distance2_centree(M,lgec):
    M2 = []
    for i in range(len(M)):
        if M[i][0] in lgec :
            M2.append(M[i])
    return M2

M2 = distance2_centree(M,lgec)

liste = [2130.1089174030517, 2051.088003962775,…]

l = sorted(liste)
```

```
x=1
L=[4,2,3]

def val_proche(x,L):
    min=10000
    val = (0,0)
    for i in range(len(L)):
        if abs(L[i]-x) <= min :
            min = abs(L[i]-x)
            val = L[i]
    return val


g = 20


def parcours(M2,l,g):
    candidates =[]
    for i in range(len(M2)):
        L = sorted(M2[i][1])
        nbr = 0
        for a in range(20):
            coef = 0.5+0.05*a
            lst = l
            pts = 0
            for j in range(10):
                dist = L[j]
                val = val_proche(coef*dist,lst)
                if abs(val-dist*coef) <= g :
                    pts = pts+1
            if pts >= nbr :
                nbr = pts
        candidates.append((M2[i][0],nbr))
    m = 0 , 0
    for b in range(len(candidates)):
        if candidates[b][1] >= m[1] :
            m = b , candidates[b][1]
    return candidates[m[0]]

etoile = parcours(M2,l,g)

print("etoile = ",parcours(M2,l,g))
```

```
def rep(photo,L_cara,etoile):
    cols,rows = photo.size
    black = (0,0,0)
    white = (255,255,255)
    red = (255,0,0)
    orange = (255,125,0)
    yellow = (255,255,0)
    green = (0,255,0)
    blue = (0,255,255)
    im = Image.new('RGBA',(cols+30,rows+30),black)
    for i in range(len(L_cara)):
        if L_cara[i][2] == "petite" :
            x = L_cara[i][0][0]
            y = L_cara[i][0][1]
            for a in range(20):
                for b in range(20):
                    im.putpixel((y-10+a,x-10+b),red)
        if L_cara[i][2] == "moyenne" :
            x = L_cara[i][0][0]
            y = L_cara[i][0][1]
            for a in range(30):
                for b in range(30):
                    im.putpixel((y-15+a,x-15+b),red)
        if L_cara[i][2] == "grosse" :
            x = L_cara[i][0][0]
            y = L_cara[i][0][1]
            for a in range(30):
                for b in range(30):
                    im.putpixel((y-15+a,x-15+b),white)
    x = etoile[0][0][0]
    y = etoile[0][0][1]
    for a in range(50):
            for b in range(50):
                im.putpixel((y-25+a,x-25+b),green)

    return im

repre = rep(photo,L_cara,etoile)

repre.show()

def latitude(etoile):
    cols,rows = photo.size
    h = etoile[0][0][0]
    d = rows - h
    rapport = d/rows
    return 71*rapport+9.68

print("latitude = ",latitude(etoile))
```

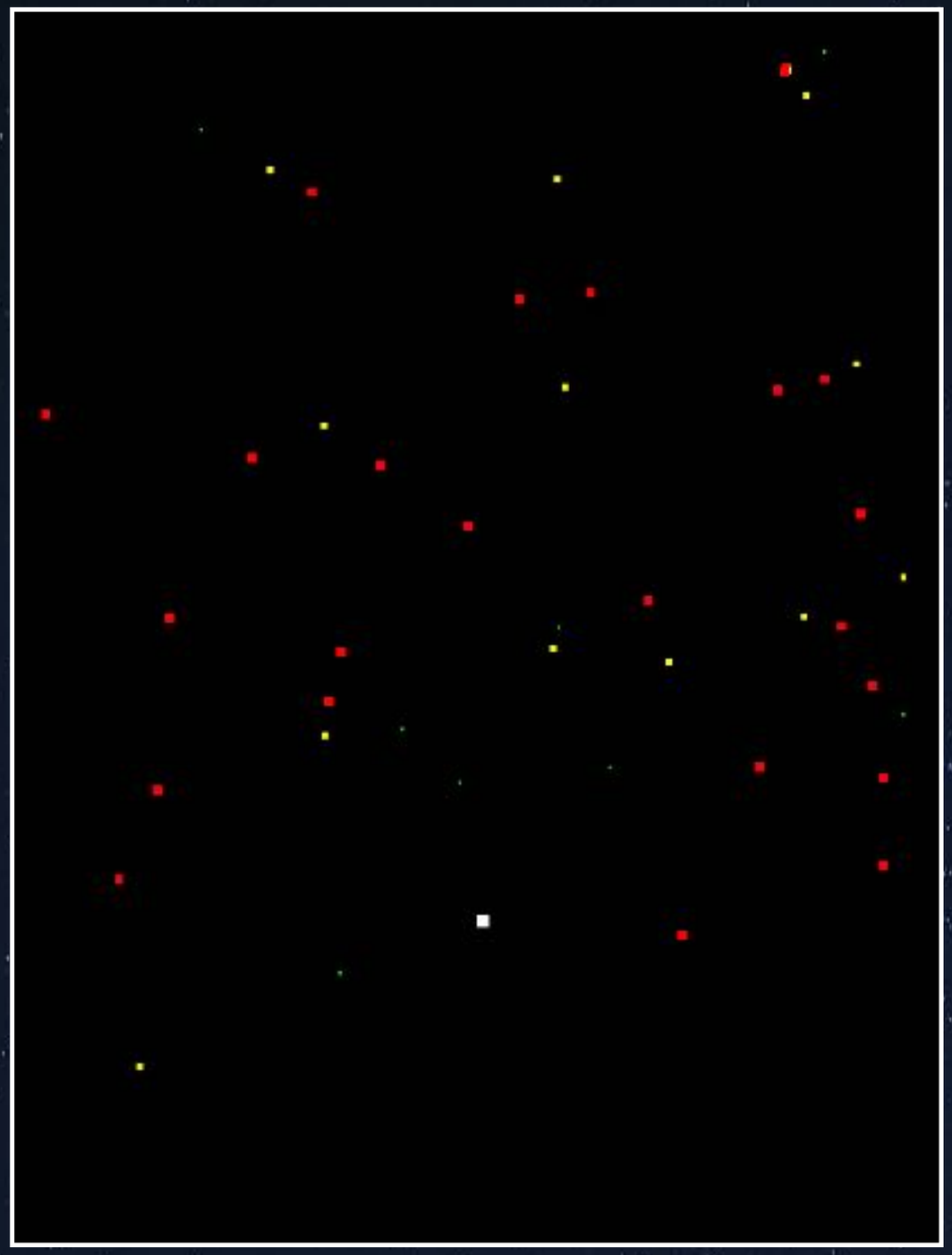


Mesure de l’angle d’ouverture d’un smartphone

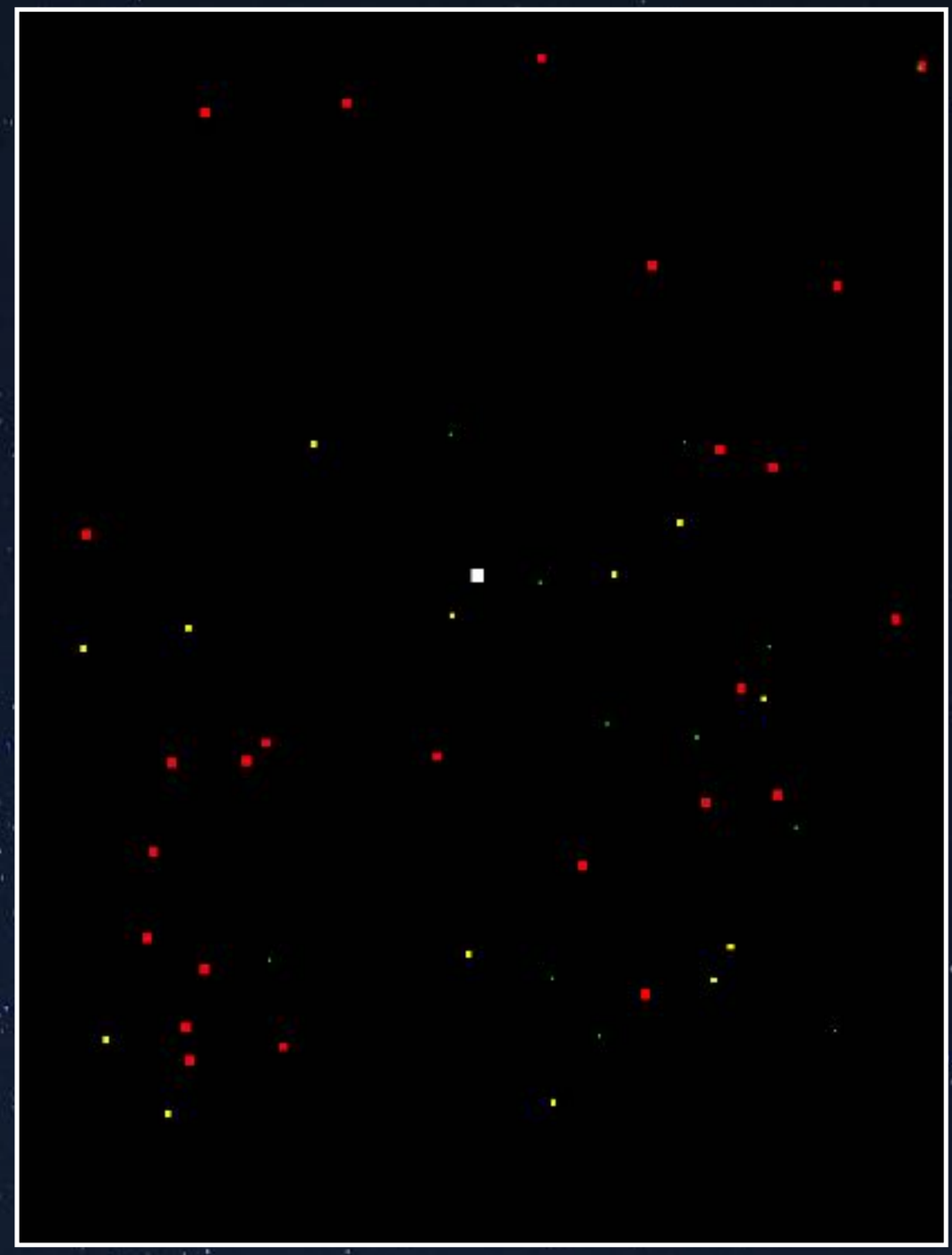
Quelques pistes pour déterminer la longitude :

→ La position des étoiles autour de l'étoile polaire à t fixé ne dépend que de la longitude.

→ Comparaison avec une référence ?



Miami (26°N)



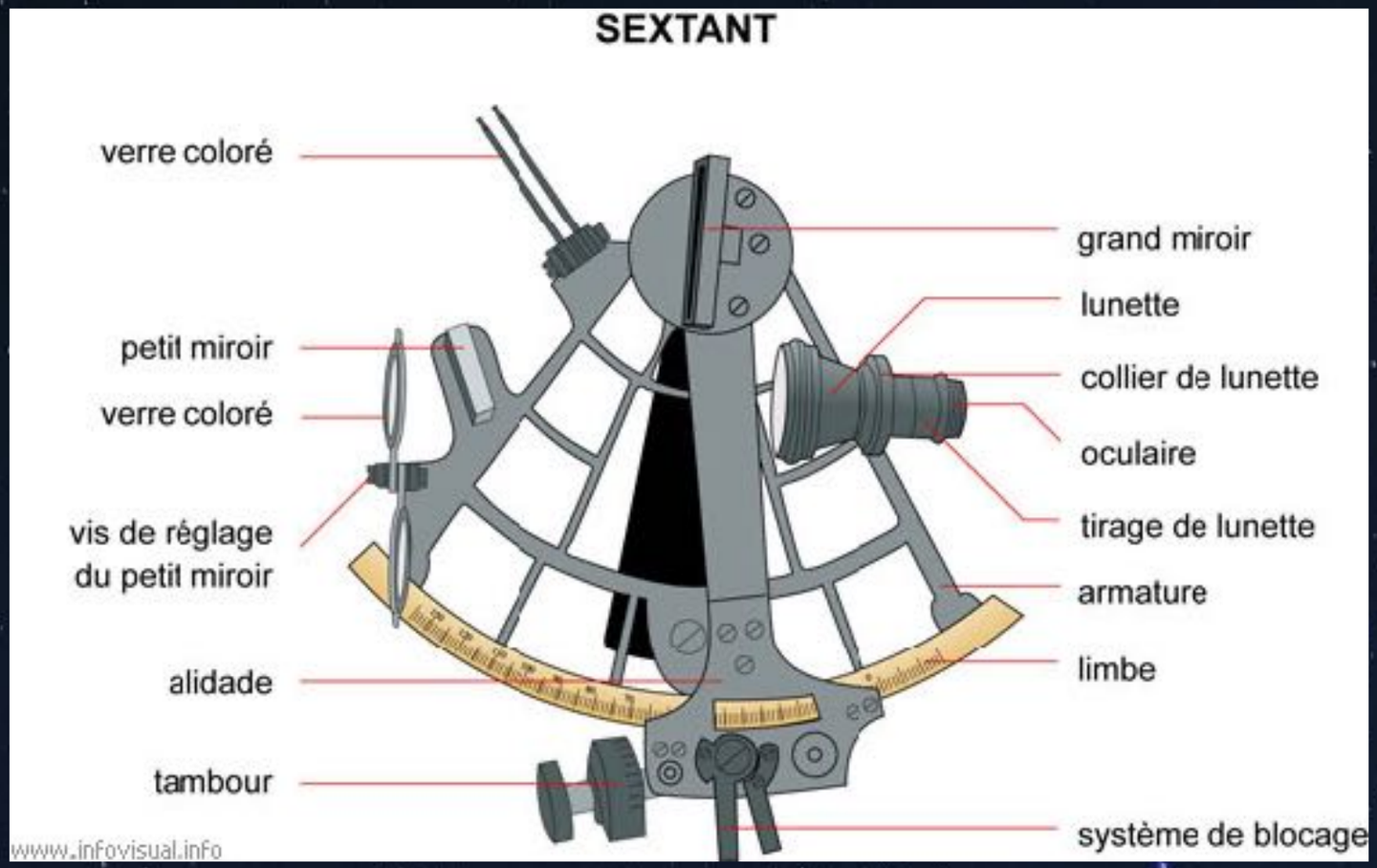
Nantes (48°N)

```
>>> %Run TIPE3.py

(3024, 4032)
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
[]
[((139, 2668), 13, 'petite'), ((191, 2543), 521, 'grosse'), ((201, 254 …
M2 =  [((((1699, 1495), 79, 'grosse'), [1836.4008277061955, 1821.482912354656 …
etoile =  (((2994, 1546), 173, 'grosse'), 7)
latitude =  27.958273809523806
```

Miami (26°N)

```
>>> %Run TIPE3.py

(3024, 4032)
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
[]
[((160, 1723), 300, 'grosse'), ((188, 2973), 302, 'grosse'), ((195, 29 …
M2 =  [((((1860, 1514), 196, 'grosse'), [1712.7991709479545, 2219.06849826678 …
etoile =  (((1860, 1514), 196, 'grosse'), 8)
latitude =  47.92702380952381
```

Nantes (48°N)



Source Image : https://bdumas.fr/AURIOL/AIDES/THEME-Transports/PLANCHE TRANSPORT-Format HTML/www.infovisual.info/05/076_fr.html

Fonctionnement du sextant

ANGLE	DISTANCE
360°	40000 KM (PÉRIMÈTRE TERRE)
1°	110 KM

Incertitude sur la latitude

Système de positionnement satellite :

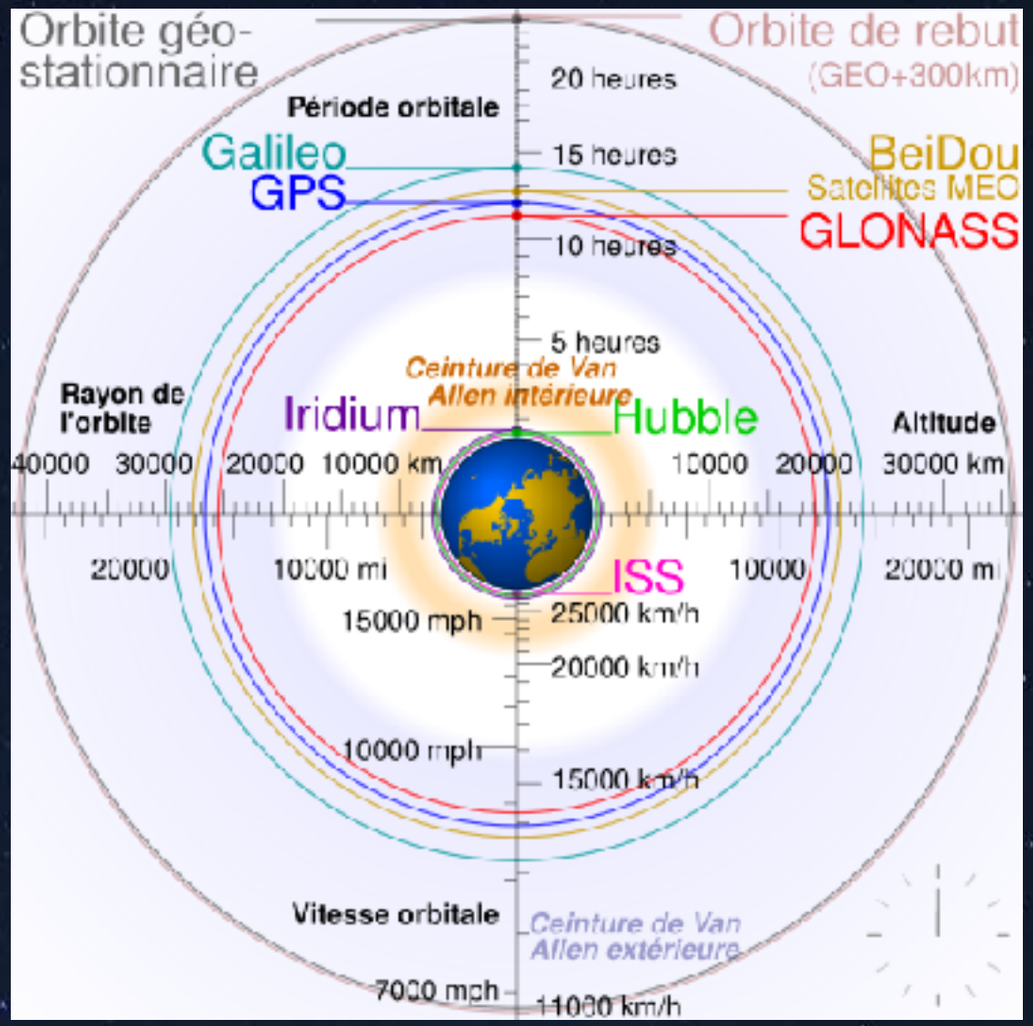
Fonctionne grâce à 140 satellites en orbite à 20000 Km d'altitude en moyenne

GPS (USA) : 78 satellites

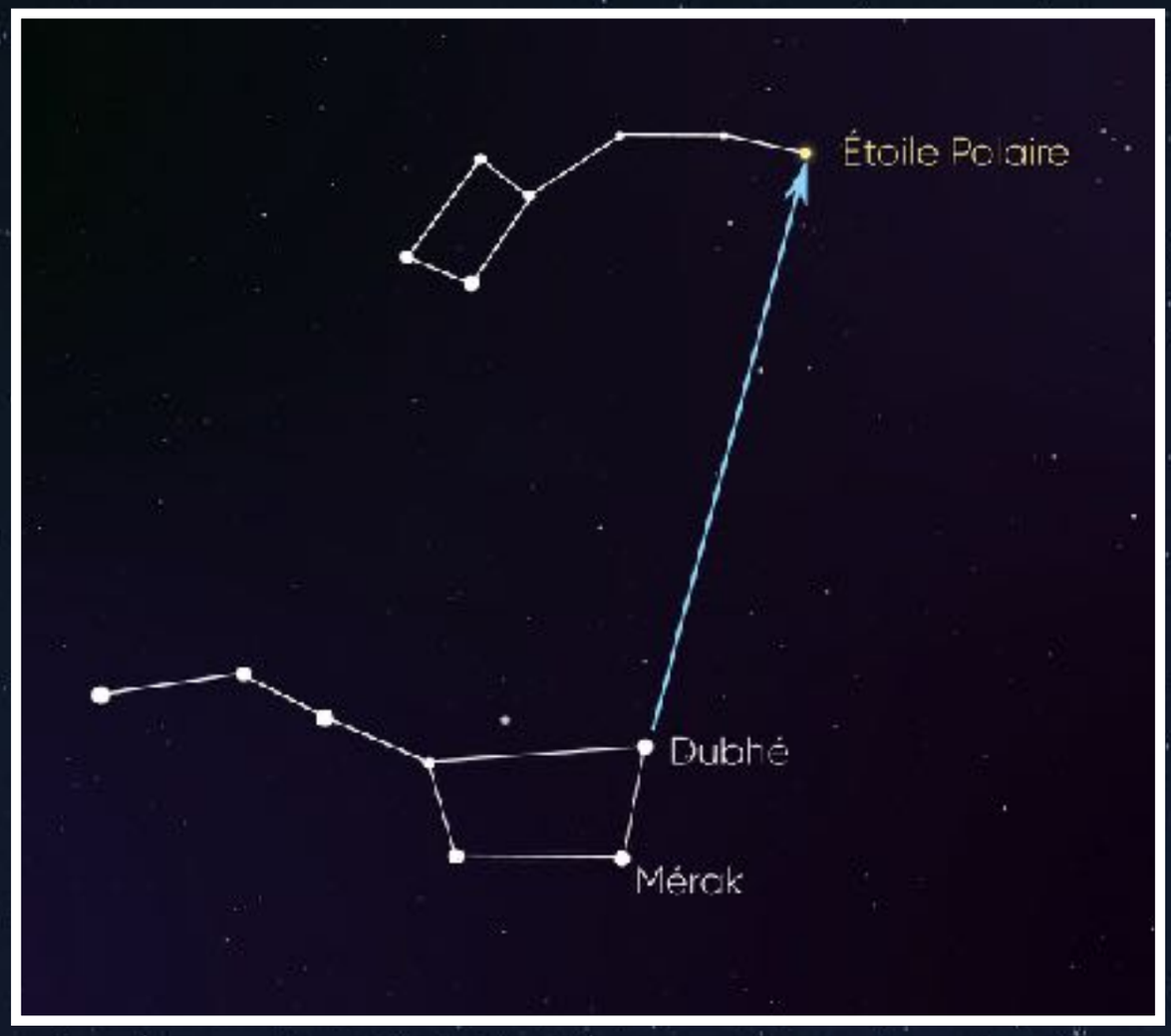
GLONASS (Russie) : 24 satellites

GALILEO (Europe) : 30 satellites

SEIDOU (Chine) : 8 satellites



Source Image : https://fr.wikipedia.org/wiki/Global_Positioning_System



Source Image : https://www.numerama.com/sciences/1609336-quest-ce-que-letoile-polaire.html

Étoile Polaire

α - Ursae Minoris
Distance : 450 AL
Constellation : Petite Ourse

Source Image : https://www.istockphoto.com/fr/photos/croix-du-sud-constellation

Croix du Sud