Détection et prévention des mélanomes dit à risque

Pauline Séret n45330

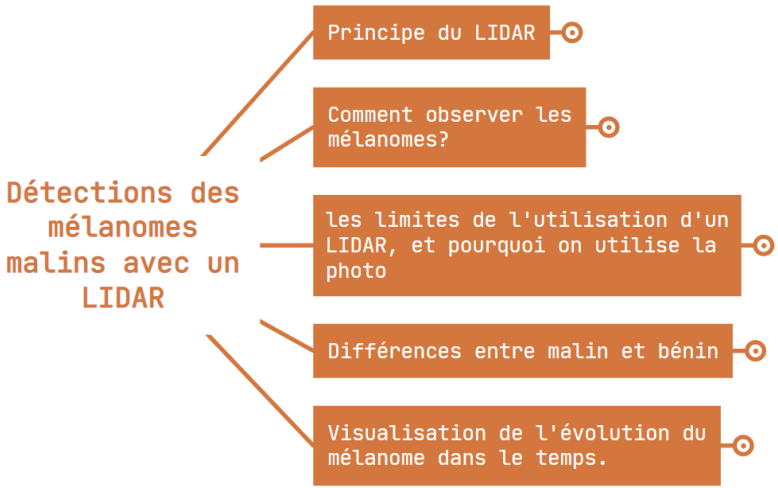
2021-2022

Introduction

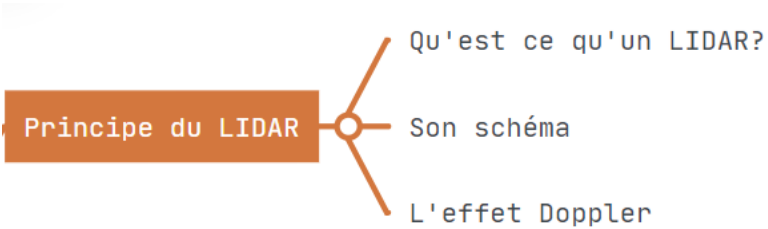
Problématique

Comment prévenir/anticiper une maladie de la peau avec les nouvelles technologies?

Plan


Détections des mélanomes malins avec un LIDAR
Principe du LIDAR
Comment observer les mélanomes?
les limites de l'utilisation d'un LIDAR, et pourquoi on utilise la photo
Différences entre malin et bénin
Visualisation de l'évolution du mélanome dans le temps.

Le Principe du LIDAR


Principe du LIDAR
Qu'est ce qu'un LIDAR?
Son schéma
L'effet Doppler

Qu'est ce qu'un LIDAR?

L.I.D.A.R.: Light Detection And Ranging

Technique de mesure électronique de distance permettant de mesurer la distance entre la surface d'un objet et l'appareil grâce aux propriétés des faisceaux de lumière.

VL6180

LIDAR commercial

Schéma

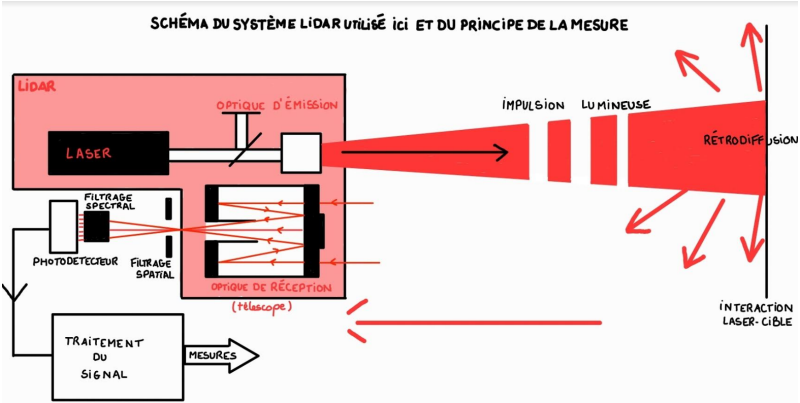
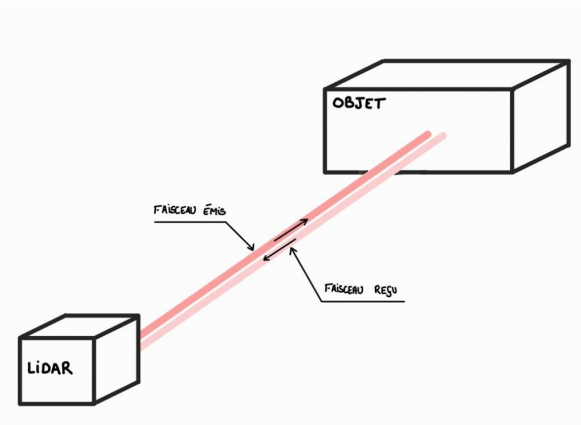
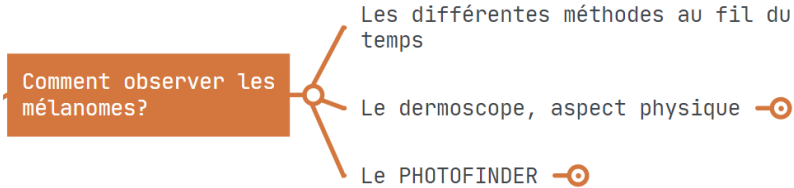


SCHÉMA DU SYSTÈME LIDAR UTILISÉ ICI ET DU PRINCIPE DE LA MESURE
LIDAR
OPTIQUE D'ÉMISSION
LASER
IMPULSION LUMINEUSE
RÉTRODIFFUSION
FILTRAGE SPECTRAL
PHOTODETECTEUR
FILTRAGE SPATIAL
OPTIQUE DE RÉCEPTION
(télescope)
INTERACTION LASER-CIBLE
TRAITEMENT DU SIGNAL
MESURES

L’effet Doppler dans le LIDAR



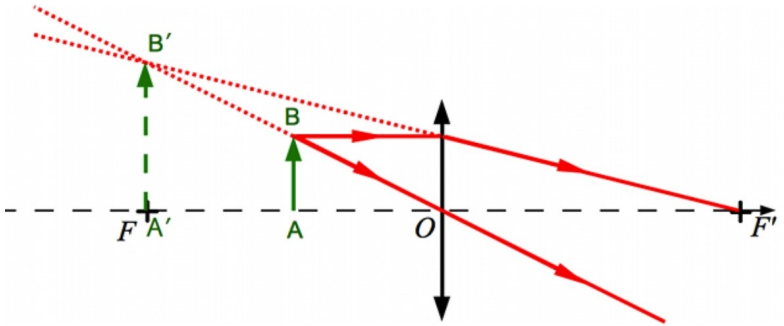
Comment observer les mélanomes?


Comment observer les mélanomes?
Les différentes méthodes au fil du temps
Le dermoscope, aspect physique
Le PHOTOFINDER

Les différentes méthodes au fils du temps

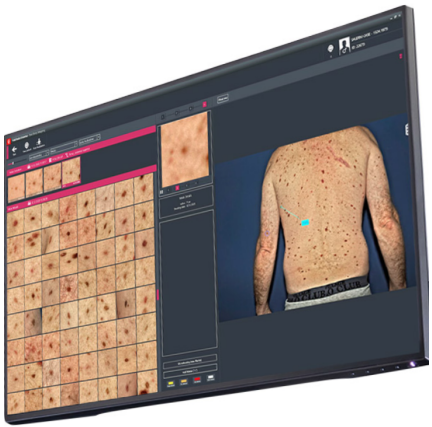
Le dermoscope, aspect physique

Image à travers une loupe



Crédit : ASM/B. Mollier

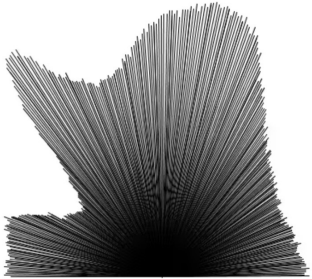
Le FOTOFINDER



Les limites d’utilisations d’un LIDAR dans ce contexte

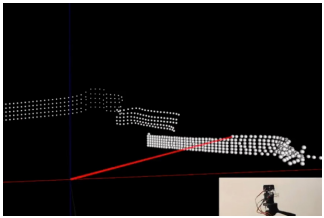
Comparaison entre les résultats

Les résultats qu'un LIDAR peut apporter.



LIDAR monté sur servomoteur

VL6180



Détection 3D d'une pièce

Différences entre malin et bénin


Différences entre malin et bénin
Le score CASH
L'ABCD'air du dermatologue

Le score CASH

Score CASH

C = Couleurs: 1 point pour chaque couleur:

A = Architecture

0 = ordonnée, 1 = un peu désordonnée , 2 = désordonnée

S = Symetrie

0 = deux axes, 1 = un seul axe, 2 = pas de symétrie

H = Homogénéité / Hétérogénéité 1 point pour chaque :

réseau | images de régression | points/
globules | stries / pseudopodes | taches
patron vasculaire polymorphe | voile gris-bleu

(Homogénéité = 1 structure seulement Hétérogénéité = 2 ou plus)

TCS = *Total CASH Score* (de 2 à 17)

(si ≥ 8 sensibilité de 98% et spécificité de 68% pour le diagnostic de mélanome)

Algorithme C.A.S.H.

C = Couleurs peu *VS.* beaucoup

A = Architecture regulière *VS.* irregulière

S = Symetrie *VS.* asymetrie

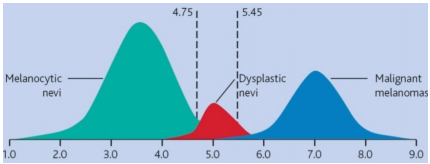
H = Homogénéité *VS.* Hétérogénéité

L'ABCD'air du dermatologue

Score ABCD *(Total dermoscopy score)*

(Stolz et al.1992)

Asymetrie	X **1,3**
+	
Bords abrupts	X **0.1**
+	
Couleurs	X **0.5**
+	
structures Dermoscopiques	X **0.5**



Visualisation de l’évolution du mélanome dans le temps

Visualisation de l'évolution du mélanome dans le temps.

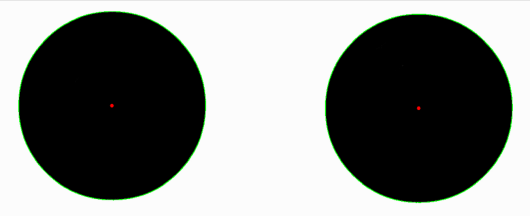
Traitement d'image avec Python

Traitement d'image

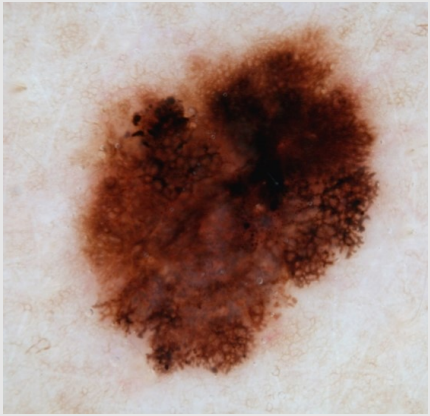
Annalogie avec des matrices

```
>>> comparaison(10)
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
[[0 0 0 0 0 0 0 0 1 1]
 [0 0 1 0 1 1 0 0 0 0]
 [1 0 0 0 0 0 0 1 0 0]
 [1 0 0 1 0 0 0 0 1 0]
 [0 0 1 0 0 1 0 0 0 0]
 [1 0 0 0 0 0 0 0 1 0]
 [1 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 1 1 0 0 0]
 [0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 1 0 0 1 0]]
```

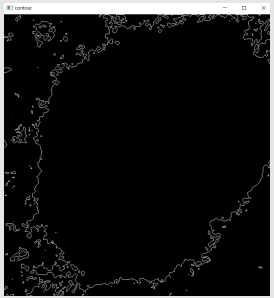
Test 1 avec des cercles

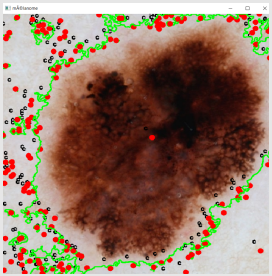


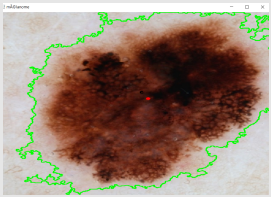
Test 2: donnés

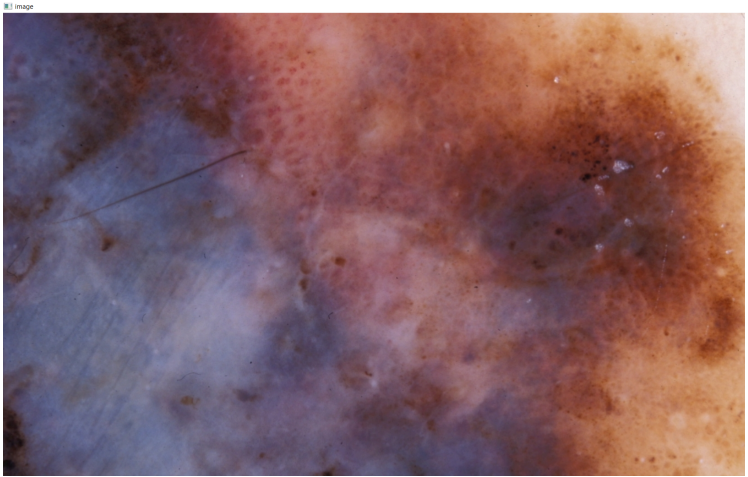


Test 2: résultats





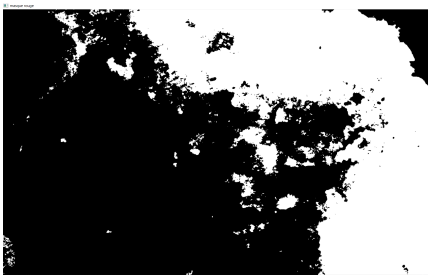




image

contour du mélanome

Bleu: min(80, 25, 20); max(130, 255, 255)



Rouge: min(0, 70, 20); max(15, 255, 255)

Marron: min(10, 25, 20); max(25, 255, 255)

Blanc: min(0, 0, 20); max(170, 0, 255)

Noir: min(80, 25, 0) max(130, 255, 0)

Conclusion

Comment prévenir/anticiper une maladie de la peau avec les nouvelles technologies?

Annexes

Annexe 1

Démonstration effet Doppler:

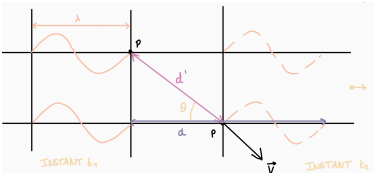
On pose $t_2 - t_1 = \frac{1}{f_p}$

On a: $d = d'.cos(\theta) + \lambda$ et $\vec{V}.\vec{u} = V * cos(\theta)$

De plus, $d = \frac{c}{f_p}$ et $\lambda = \frac{c}{f_e}$

Donc, $\frac{c}{f_p} = \frac{(\vec{V}.\vec{u})}{f_p} + \frac{c}{f_e}$

D'où, $\vec{V}.\vec{u} = c.(1 - \frac{f_p}{f_e})$



Annexe 2

```python
import numpy as np
import random

def comparaison(10):
    A = np.zeros((10, 10))
    A[10-9:10-8, 0:10] = A[10-7:10-6, 0:10] = A[10-5:10-4, 0:10] = A[10-3:10-2, 0:10] = A[10-1:10, 0:10] = 1
    B = np.random.choice([1, 0], size=[10,10], p=[.2, .8])
    print(A)
    print(B)
    if B == A:
        return 'Les deux matrices sont différentes, il faut donc regarder ce qu il y a de différents'
    else:
        return 'Les deux matrices sont identiques, on ne fait rien'
```

Annexe 3

```
edged = cv2.Canny(gray, 30, 200)
cv2.waitKey(0)

# trouver les contours
# utilisation de copy parce que faire le contour altère l'image
contours, hierarchy = cv2.findContours(edged, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

cv2.imshow('Canny Edges After Contouring', edged)
cv2.waitKey(0)

print("Number of Contours found = " + str(len(contours)))

#dessiner les contours
# -1 signifies drawing all contours
cv2.drawContours(image, contours, -1, (0, 255, 0), 3)

cv2.imshow('Contours', image)
cv2.waitKey(0)
cv2.destroyAllWindows(_
```

```
import cv2
import numpy as np
image = cv2.imread('image2.jpg')
cv2.waitKey(0)
# niveau de gris
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

edged = cv2.Canny(gray, 30, 200)
cv2.waitKey(0)
# trouver les contours
# utilisation de copy parce que faire le contour altère l'image
contours, hierarchy = cv2.findContours(edged, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

cv2.imshow('Canny Edges After Contouring', edged)
cv2.waitKey(0)
print("Number of Contours found = " + str(len(contours)))
#dessiner les contours
# -1 signifies drawing all contours
cv2.drawContours(image, contours, -1, (0, 255, 0), 3)

cv2.imshow('Contours', image)
cv2.waitKey(0)
cv2.destroyAllWindows(_
```

Annexe 4

```python
import cv2 as cv
import numpy as np

image = cv.imread("image2.jpg")
gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

blur = cv.GaussianBlur(gray, (5, 5), cv.BORDER_DEFAULT)
ret, thresh = cv.threshold(blur, 200, 255, cv.THRESH_BINARY_INV)

cv.imwrite("thresh.png", thresh)

contours_hierarchies = cv.findContours(thresh, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)

blank = np.zeros(thresh.shape[:2], dtype='uint8')

cv.drawContours(blank, contours, -1, (255, 0, 0), 1)

cv.imwrite("contour.png", blank)
cv.imshow("contour", blank)
cv.waitKey(0)
```

```python
for i in contours:
    M = cv.moments(i)
    if M['m00'] != 0:
        cx = int(M['m10'] / M['m00'])
        cy = int(M['m01'] / M['m00'])
        cv.drawContours(image, [i], -1, (0, 255, 0), 2)
        cv.circle(image, (cx, cy), 7, (0, 0, 255), -1)
        cv.putText(image, "center", (cx - 20, cy - 20),
                   cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2)
    print(f"x: {cx} y: {cy}")

im = cv.imwrite("image.png", image)
cv.namedWindow("image", cv.WINDOW_NORMAL)
cv.imshow("image", image)
cv.waitKey(0)
```

Annexe 5

```python
import cv2 as cv
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

image = cv.imread("image2.jpg")
gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

blur = cv.GaussianBlur(gray, (5, 5), cv.BORDER_DEFAULT)
ret, thresh = cv.threshold(blur, 200, 255, cv.THRESH_BINARY_INV)

cv.imwrite("thresh.png", thresh)

contours_hierarchies = cv.findContours(thresh, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)

blank = np.zeros(thresh.shape[:2], dtype='uint8')

cv.drawContours(blank, contours, -1, (255, 0, 0), 1)

cv.imwrite("contour.png", blank)
cv.imshow("contour", blank)
cv.waitKey(0)
```

```python
for i in contours:
    M = cv.moments(i)
    if M['m00'] != 0:
        cx = int(M['m10'] / M['m00'])
        cy = int(M['m01'] / M['m00'])
        cv.drawContours(image, [i], -1, (0, 255, 0), 2)
        cv.circle(image, (cx, cy), 7, (0, 0, 255), -1)
        cv.putText(image, "center", (cx - 20, cy - 20),
                   cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2)
    print(f"x: {cx} y: {cy}")

im = cv.imwrite("image.png", image)
cv.namedWindow("image", cv.WINDOW_NORMAL)
cv.imshow("image", image)
cv.waitKey(0)
```

Annexe 6

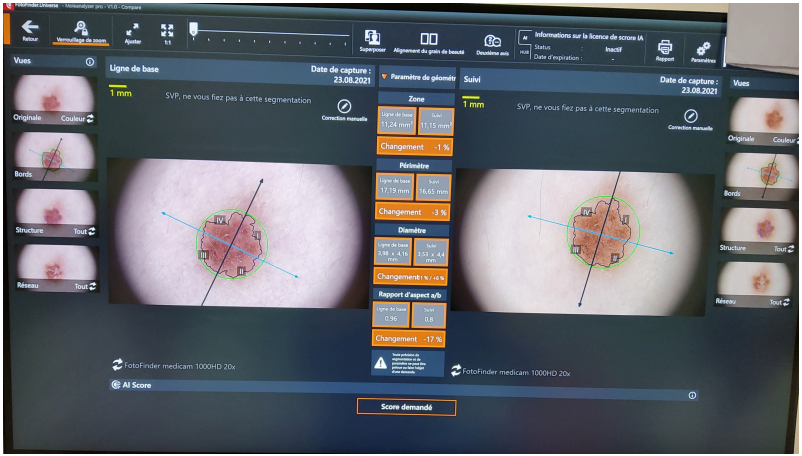
```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import image


img = cv.imread("phtoto1.jpg")

hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)

lower_range_bleu = np.array([80, 25, 20])
upper_range_bleu = np.array([130, 255, 255])

lower_range_rouge = np.array([0, 70, 20])
upper_range_rouge = np.array([15, 255, 255])

lower_range_marron = np.array([10, 25, 20])
upper_range_marron = np.array([25, 255, 255])

lower_range_blanc = np.array([0, 0, 20])
upper_range_blanc = np.array([170, 0, 255])

lower_range_noir = np.array([80, 25, 0])
upper_range_noir = np.array([130, 255, 0])
```

```
maskbleu = cv.inRange(hsv, lower_range_bleu, upper_range_bleu)
maskrouge = cv.inRange(hsv, lower_range_rouge, upper_range_rouge)
maskmarron = cv.inRange(hsv, lower_range_marron, upper_range_marron)
masknoir = cv.inRange(hsv, lower_range_blanc, upper_range_blanc)
maskblanc = cv.inRange(hsv, lower_range_noir, upper_range_noir)

cv.imshow("image", img)
cv.imshow("masque bleu", maskbleu)
cv.imshow("masque rouge", maskrouge)
cv.imshow("masque marron", maskmarron)
cv.imshow("masque blanc", maskblanc)
cv.imshow("masque noir", masknoir)

cv.waitKey(0)
cv.destroyAllWindows()
```

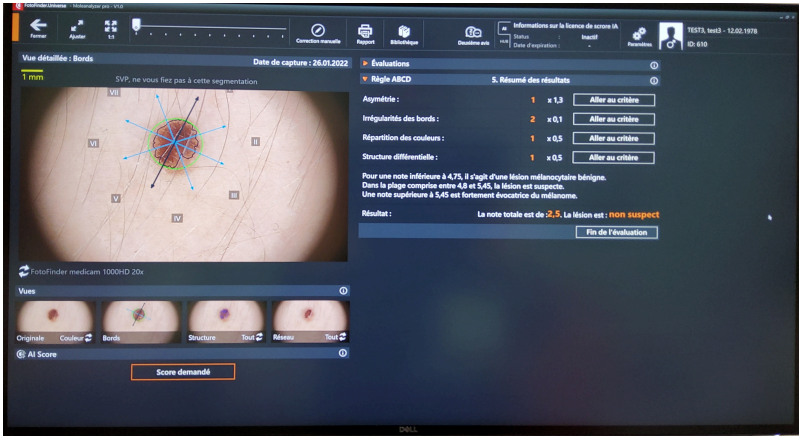
Annexe 7.1
Photos d'un Fotofinder

Annexe 7.2

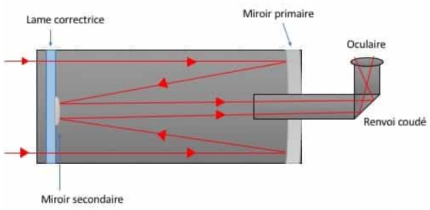
Annexe 7.3


Retour
Vérrouillage de zoom
Ajuster
1:1
Superposer
Alignement du grain de beauté
Deuxième avis
Informations sur la licence de scrore IA
Status
Inactif
Date d'expiration :
Rapport
Paramètres
Vues
Originale
Couleur
Bords
Structure
Tout
Réseau
Tout
Ligne de base
Date de capture :
23.08.2021
1 mm
SVP, ne vous fiez pas à cette segmentation
Correction manuelle
Paramètre de géométr
Zone
Ligne de base
11,24 mm²
Suivi
11,15 mm²
Changement -1 %
Périmètre
Ligne de base
17,19 mm
Suivi
16,65 mm
Changement -3 %
Diamètre
Ligne de base
3,98 x 4,16 mm
Suivi
3,53 x 4,4 mm
Changement
Rapport d'aspect a/b
Ligne de base
0,96
Suivi
0,8
Changement -17 %
Suivi
Date de capture :
23.08.2021
1 mm
SVP, ne vous fiez pas à cette segmentation
Correction manuelle
Vues
Originale
Couleur
Bords
Structure
Tout
Réseau
Tout
FotoFinder medicam 1000HD 20x
FotoFinder medicam 1000HD 20x
AI Score
Score demandé

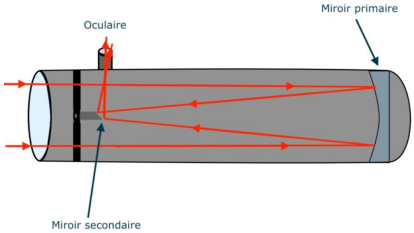
Annexe 7.4


Fermer
Ajouter
Correction manuelle
Rapport
Bibliothèque
Deuxième avis
Informations sur la licence de scrore IA
Status : Inactif
Date d'expiration :
Paramètres
TEST3, test3 - 12.02.1978
ID: 610
Vue détaillée : Bords
Date de capture : 26.01.2022
1 mm
SVP, ne vous fiez pas à cette segmentation
VII
VI
III
V
IV
FotoFinder medicam 1000HD 20x
Vues
Orginale
Couleur
Bords
Structure
Tout
Réseau
Tout
AI Score
Score demandé
Évaluations
Règle ABCD
5. Résumé des résultats
Asymétrie : 1 x 1,3 Aller au critère
Irrégularités des bords : 2 x 0,1 Aller au critère
Répartition des couleurs : 1 x 0,5 Aller au critère
Structure différentielle : 1 x 0,5 Aller au critère
Pour une note inférieure à 4,75, il s'agit d'une lésion mélanocytaire bénigne.
Dans la plage comprise entre 4,8 et 5,45, la lésion est suspecte.
Une note supérieure à 5,45 est fortement évocatrice du mélanome.
Résultat : La note totale est de :2,5. La lésion est : non suspect
Fin de l'évaluation
DELL

Annexe 8

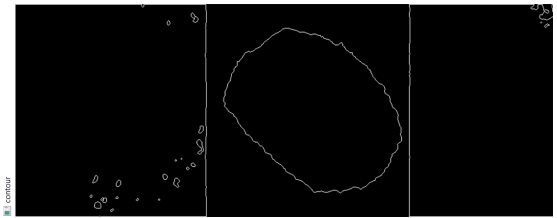


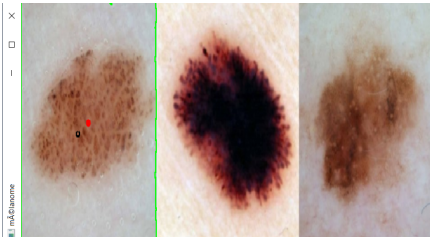
Téléscope Cassegrin



Téléscope Newton

Annexe 9





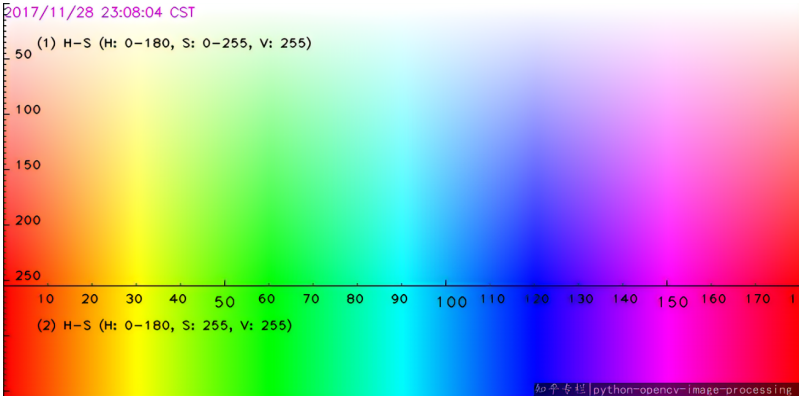
Annexe 10

SparkFun ToF Range Finder Sensor - VL6180

Feature	Detail
Package	Optical LGA12
Size	4.8 x 2.8 x 1.0 mm
Ranging	0 to 100 mm[1]
Ambient light sensor	< 1 Lux up to 100 kLux[2] 16-bit output[3] 8 manual gain settings
Operating voltage: • Functional range • Optimum range[4]	 2.6 to 3.0 V 2.7 to 2.9 V
Operating temperature: • Functional range • Optimum range[4]	 -20 to 70°C -10 to 60°C
Typical power consumption	Hardware standby (GPIO0 = 0): < 1 µA Software standby: < 1 µA ALS: 300 µA Ranging: 1.7 mA (typical average)[5]
IR emitter	850 nm
I^2C	400 kHz serial bus Address: 0x29 (7-bit)

Datasheet LIDAR

Annexe 11


2017/11/28 23:08:04 CST
(1) H-S (H: 0-180, S: 0-255, V: 255)
50
100
150
200
250
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170
(2) H-S (H: 0-180, S: 255, V: 255)

Annexe 12

```python
import cv2 as cv
import numpy as np

image = cv.imread("3melanomes.jpg")
gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

blur = cv.GaussianBlur(gray, (5, 5), cv.BORDER_DEFAULT)
ret, thresh = cv.threshold(blur, 200, 255, cv.THRESH_BINARY_INV)

cv.imwrite("thresh.png", thresh)

contours, hierarchies = cv.findContours(thresh, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)

blank = np.zeros(thresh.shape[:2], dtype='uint8')

cv.drawContours(blank, contours, -1, (255, 0, 0), 1)

cv.imwrite("contour.png", blank)
cv.imshow("contour", blank)
cv.waitKey(0)
```

```python
areas = [cv.contourArea(i) for i in contours]
max_index = np.argmax(areas)
cnt = contours[max_index]
print(max_index)
cv.drawContours(image, contours, max_index, (0, 255, 0), 2)

for i in contours:
    M = cv.moments(cnt)
    if M['m00'] != 0:
        cx = int(M['m10'] / M['m00'])
        cy = int(M['m01'] / M['m00'])

cv.circle(image, (cx, cy), 5, (0, 0, 255), -1)
cv.putText(image, "c", (cx - 20, cy - 20),
    cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2)



im = cv.imwrite("melanome.png", image)
cv.namedWindow("mélanome", cv.WINDOW_NORMAL)
cv.imshow("mélanome", image)
cv.waitKey(0)
```