University Paris-Est Creteil, Creteil, France

Master 1 OIVM Optique Image Vision MultiMedia

ATDN II

Auteur:
Nassim Battache
Iskander Djouad

Superviseur :
Corinne lagorre

July 19, 2022

Abstract

le morphing ou la morphose est un concept très connu dans le monde du traitement d'image, inventer en 1982 par TOM BRIGHAM. Ce dernier consiste à générer des images ou autrement dit des animations qui transforme une image initiale a une image finale d'une façon la plus naturel et la plus fluide possible. L'image initiale et finale sont choisi d'une manière arbitraire par l'utilisateur.

Ce rapport comporte une approche dite géométrique du concept de mouvement d'image avec les 3 type d'images (binaire , gris , couleurs).

L'algorithme développer génère des vidéos illustrons le passage d'un tracer initial (forme initial) vers un tracer final (forme final) autrement dit chercher une forme médiane.

outils de développement : python 3.8.5 (skimage , numpay , matplotlib,imageio)

Environnemt : Spyder

Contents

Introduction

Ce laboratoire manipule trois types d'images, quel sont ces trois types ? c'est quoi la différence entre ces dernières ?

0.0.1 Image binaire :

Commencent par la plus simple qui est l'image binaire. une image binaire est une image numérique et matricielle, cette dernière peut avoir que deux couleur le noir et le blanc, Donc coder sur 1 bit par pixel ou les cases de cette matrice peuvent prendre soit la valeur 0 ou 1.

Figure 1: Image binaire [4]

0.0.2 Images en niveau de gris :

Les images en niveau de gris sont des images qui sont coder sur 8 bits d'une façon ou chaque pixel peut avoir une valeur allant de 0 à 255 ou le 0 correspond à la couleur noire et le 255 a la couleur blanche.

Figure 2: Image en niveau de gris [4]

0.0.3 Images en couleurs :

Le dernier cas ce sont les images en couleurs, c'est le type le plus utiliser dans les smartphones les écrans...

ce genre d'image et composer de 3 canons qui sont le vert, bleu et le rouge (RVB) ou chaque canon a 256 nuances comme dans le cas des images en niveau de gris, autrement dit chaque pixel d'un canon est codé sur 8 bits ce qui fait que chaque pixel de l'image en couleur sera codé sur 24 bits.

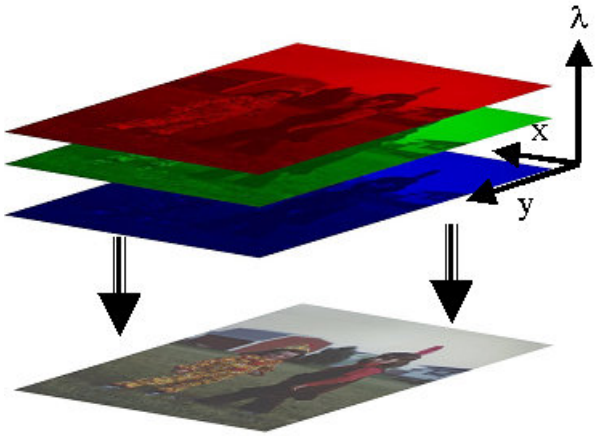


Figure 3: Image en couleur [4]

Chapter 1

pré-traitement :

1.0.0.0.1 cette étape consiste a créez une matrice de transformation, qui est un tableau 2D.

Cette matrice contient les informations nécessaires pour décaler l'image,selon les axes x et y .

il est nécessaire d'extraire les regionproprs a chaque forme des deux images a fin de les décaler (dans notre cas les centrer) .

Remarque : cette procédure est tester uniquement les images binaire .

```
  selon les axes x et y.
def translation_img(src_img,shift_distance, shape_of_out_img):
    h,w = src_img.shape[:2]
    out_img=np.zeros(shape_of_out_img,src_img.dtype)
    for i in range(h):
        for j in range(w):
            new_x=j+shift_distance[0]
            new_y=i+shift_distance[1]
            if 0<=new_x<w and 0<=new_y<h: # Test d'écriture
                out_img[new_y,new_x]=src_img[i,j]
    return out_img
```

```
IN1=imread('i1.png',as_gray=True)
IN2=imread('i2.png',as_gray=True)

# Conversion en images binaires GRAY -> BOOLEAN
IM1=(IN1==0) # transformer l'image gris en une image noir et blanc (binaire)
imshow(np.uint8(IM1)*255) # Pour la visu, images 8 bits
plt.show()

IM2=(IN2==0)
imshow(np.uint8(IM2)*255)
plt.show()


# Positionnement / Centrage des 2 formes

# centre de l'image
(szx,szy)=IM1.shape # creation de deux vecteur szx : pixel ligne ,szy :pixel colone
centrex=int(szx/2) #le pixel du mileux de la ligne (au mileu de l'axe x)
centrey=int(szy/2) #le pixel du mileux de la colone (au mileu de l'axe y)
#(centrex , centrey) : le pixel du mileu
# centrage Forme 1
'''
les points tels que Y(x,y,t)>0 ne correspondent pas forcément aux points mobiles
'''
(tabx, taby)=np.where(IM1==1) # les regions ou les pixel sont diffirent de null (regionproprs)
mx=int(np.sum(tabx)/np.size(tabx)) # Moyenne arithmétique sur les  de  formes
my=int(np.sum(taby)/np.size(taby)) # # Moyenne arithmétique sur les pixel de  formes

dx=int(centrex-mx) # distance equlidient entre le mileu de l'axe x et les regions de l'axe
dy=int(centrey-my) # # distance equlidient entre le mileu de l'axe y et les regions de l'axe

FORME1=translation_img(IM1,(dy,dx), IM1.shape) # centrer l'image
```

Figure 1.1: pré$_t$*raitement*

[BATTACHE Nassim, Iskander Djouad]

Chapter 2

interpolation des formes :

2.1 Morphologie mathématique :

[Djouad iskander et Battache Nassim]

2.1.1 Erosion morphologique :

L'érosion morphologique définit un pixel en (i,j) au minimum sur tous les pixels du voisinage centré en (i,j). L'érosion rétrécit les régions claires et agrandit les régions sombres. en plus L'érosion est l'opération inverse, qui est définie comme une dilatation du complémentaire de la structure. Elle consiste à chercher tous les pixels pour lesquels l'élément structurant centré sur ce pixel touche l'extérieur de la structure. Le résultat est une structure rognée. On observe la disparition des particules plus petites que l'élément structurant utilisé, et la séparation éventuelle des grosses particules.

il existe dans skimage une fonction érosion

Figure 2.1: dilatation et érosion de deux formes

2.1.2 La dilatation morphologique :

La dilatation morphologique définit la valeur d'un pixel au maximum sur toutes les valeurs de pixel dans un voisinage local centré sur lui. Les valeurs où l'empreinte est 1 définissent ce voisinage. La dilatation agrandit les régions claires et rétrécit les régions sombres. En plus une dilatation morphologique consiste à déplacer l'élément structurant sur chaque pixel de l'image, et à regarder si l'élément structurant « touche » (ou plus formellement intersecte) la structure d'intérêt. Le résultat est une structure qui plus grosse que la structure d'origine. En fonction de la taille de l'élément structurant, certaines particules peuvent se trouver connectées, et certains trous disparaître.

il existe dans skimage une fonction dilatation

2.2 l'intersection entre deux formes :

Pour cela il existe deux méthodes différentes :

• la premier consiste a calculer l'infinimum entre les deux formes , le but est de parcourir les deux images et de prendre le valeur minimal des pixels entre ces deux dernier .

• Ou bien utiliser la bibliothèque numpy qui contien la fonction minimum qui nous permet de calculer ou comparer deux tableaux et renvoie un nouveau tableau contenant les minima élément par élément.

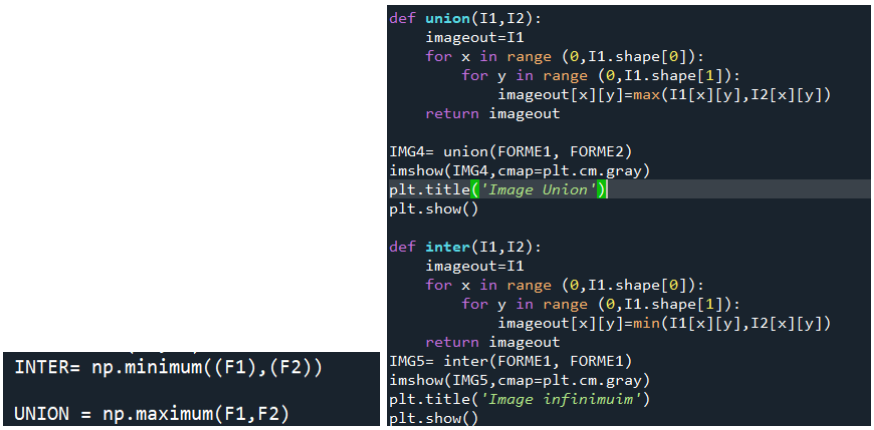
mais cette dernier ne procède pas sur des images de taille diffèrent . d'effectuer un « or » entre les deux images.

2.3 L'union de deux formes :

De la même manière que l'intersection l'union ce fait soit avec :

• np.maximum (X , Y) ou la fonction compare les deux tableaux et renvoie un tableau des minima élément par élément.

• ou bien calculer implémenter une fonction maximum qui rédimenssion les deux images prend la valuer maximum entre ces deux deriners



```
INTER= np.minimum((F1),(F2))

UNION = np.maximum(F1,F2)
```

```
def union(I1,I2):
    imageout=I1
    for x in range (0,I1.shape[0]):
        for y in range (0,I1.shape[1]):
            imageout[x][y]=max(I1[x][y],I2[x][y])
    return imageout

IMG4= union(FORME1, FORME2)
imshow(IMG4,cmap=plt.cm.gray)
plt.title('Image Union')
plt.show()

def inter(I1,I2):
    imageout=I1
    for x in range (0,I1.shape[0]):
        for y in range (0,I1.shape[1]):
            imageout[x][y]=min(I1[x][y],I2[x][y])
    return imageout
IMG5= inter(FORME1, FORME1)
imshow(IMG5,cmap=plt.cm.gray)
plt.title('Image infinimuim')
plt.show()
```

Figure 2.2: Intersection et union de deux formes

2.4 Chercher une forme médiane :

A fin de déterminer une forme médiane il est nécessaire d'implémenter une fonction qui nous permet de calculer la médiane entre deux formes . Cette médiane peut être calculer soit par SKIZ (c'est la Ligne de Partage des Eaux entre la partie commune à X et à Y et la partie extérieure aux deux ensembles réunis) ou a l'aide d'opérations morphologiques simples comme suivant :

$$M(X,Y) = IZ_{(X\cup Y)}(X\cap Y) = \bigcup_{\forall \lambda} \left\{ \left[(X\cap Y) \oplus \lambda B \right] \cap \left[(X\cup Y) \ominus \lambda B \right] \right\}$$

Figure 2.3: Formule opérations morphologiques

Dans ce lab on va plutôt utiliser la deuxième option qui est les opérations morphologiques simples ou X et Y représente les deux images de départ.

Concernant les opérateurs d'intersection et d'union on va traduire ça avec la notion de maximum, minimum et puis par érosion et dilatation.

On part de deux images X et Y

On dilate min(X,Y) (intersection de X et de Y)

On érode max(X,Y) (union de X et de Y)

On prend l'intersection entre les deux résultats

On recommence pour des tailles d'éléments structurants croissantes (jusqu'à idempotence) [slides]

Chapter 3

Poursuite de la transformation :

[djouad iskander et BATTACHE Nassim]

3.1 Mouvement des formes

Pour cela il faudrai implémenter une fonction récursive qui nous permet d'itérer la fonction médiane selon le nombres de médiane souhaité tout en insérons ces dernier dans le bon ordre autrement dit au milleu des deux forme traité .

3.2 visualisation :

Dans cette partie en vas utliser la fonction mimsave contenue dans la bibliothèque imageio a fin de sauvegardes les résultats dans un ficher .giff dans le but de visualiser la transformation .

```
def image_movment(image1,image2, taille ) :

 for k in range(taille):
    print(k) #numero d'image
    nb=0 # rang de l'image de depart
    longueur=len(Liste_mediane)-1 #longeur de la liste
    position_dep = 0
    for position_dep in range (longueur) :
        ''' calcule de la mediane '''
        me =mediane(Liste_mediane[position_dep+nb],Liste_mediane[position_dep+nb+1])
        imshow(me,cmap=plt.cm.gray)
        plt.title('Image mediane')
        plt.show()
        Liste_mediane.insert(position_dep+1+nb,me) #inserer la mediane au milleu
        nb+=1

video_gris =image_movment(IN1,IN2,5 )
Liste_mediane2 = []
for v in range(len(Liste_mediane)) :
  Liste_mediane2.append(np.uint8((Liste_mediane[v])*255))

imageio.mimsave('vid.gif',Liste_mediane,fps=10)
```

Figure 3.1: Mouvement des formes

Chapter 4

Méthode et résultat :

4.1 pour l'image binaire :

4.1.1 En enterée :

Cette étape commence par le choix des deux premières images (image de départ et l'image d'arriver) .

La taille maximal de l'élément structurent de préférence sa taille ne doit pas dépasser la taille des formes contenue dans les images. dans notre cas on a des image de taille 200*200 par Par conséquent la taille de l'élément structurant et de taille 128.

La taille de la vidéo est de : $(2^n) + 1$.

fsp(nombre d'image par seconde) : 10

4.1.2 Traitement :

binariser les deux images d'entrer .

centrer les objet de l'image.

calculer les médianes.

4.1.3 En sortie :

Après le calcul des médianes et leur insertion progressive dans la liste médiane, l'affichage de quelques éléments de cette liste et le suivant :

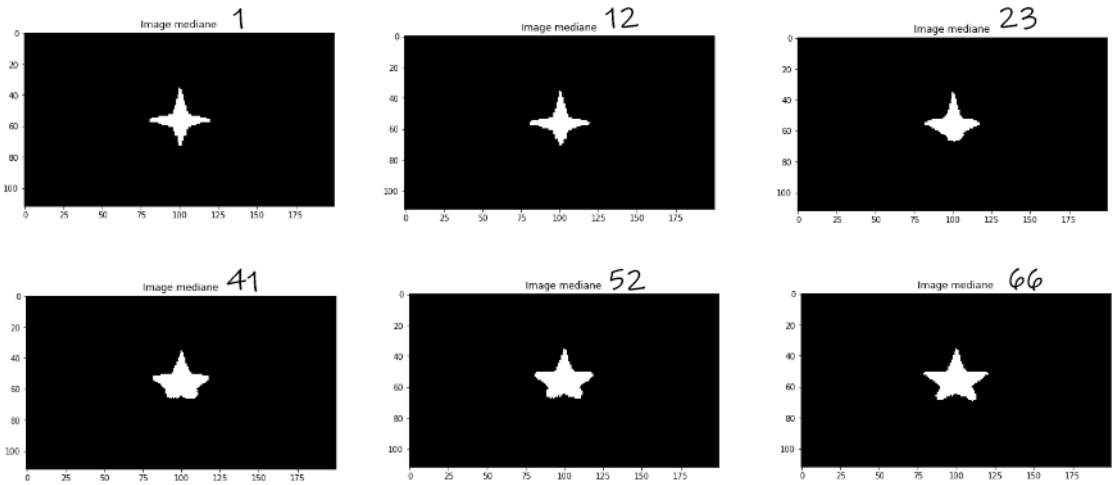


Figure 4.1: Images médiane Binaire

A partir de cette liste dés a présent l'enregistrement de vidéo est possible. [djouad iskander et BATTACHE Nassim]

4.2 pour l'image en niveau de gris :

4.2.1 En entrée :

De même cette étape commence par le choix des deux premières images (image de départ et l'image d'arriver) en couleur cette fois,

avant tout choix d'image il faut s'assurer que celles-ci ont la même taille pareil pour les autres types d'image, Dans ce cas les images sont a 200*200 pixels.

La taille maximal de l'élément structurent de préférence sa taille ne doit pas dépasser la taille des formes contenue dans les images. dans notre cas on a des image de taille 200*200 par Par conséquent la taille de l'élément structurant et de taille 128.

4.2.2 Traitement :

ici l'image contient 3 canons pour faire un traitement pour une image de niveau de gris il suffit juste de choisir le premier canon et de calculer ses médianes .

4.2.3 En sortie :

Après le calcul des médianes et leur insertion progressive dans la liste médiane, l'affichage de quelques éléments de cette liste et le suivant :

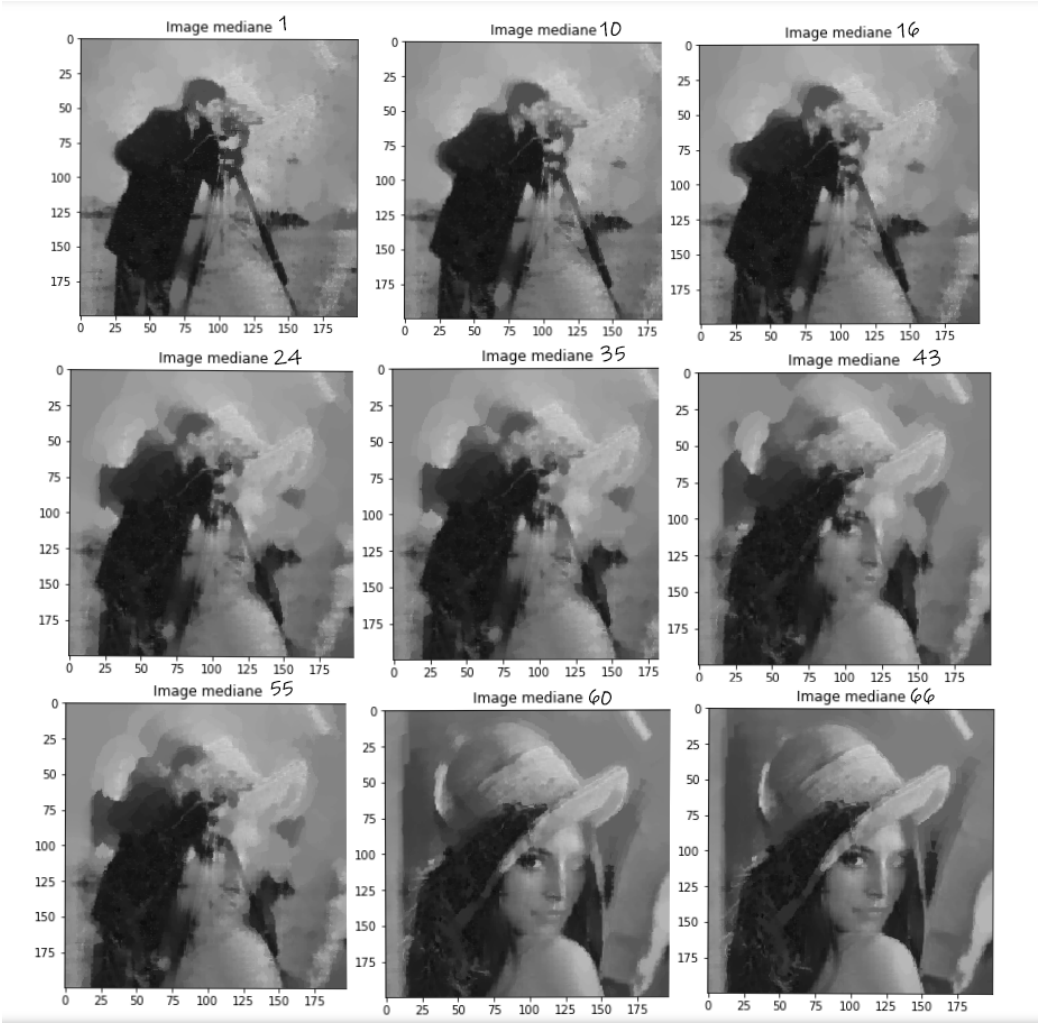


Figure 4.2: Images médiane niveau de gris

De même a partir de cette liste dés a présent l'enregistrement de vidéo est possible. [djouad iskander et BATTACHE Nassim]

4.3 pour l'image en couleur :

4.3.1 En entrée :

Dans cette partie comporte uniquement le choix des deux images en couleur de départ.

4.3.2 Traitement :

pour cette partie c'est presque le même traitement que l'image en niveau de gris par contre cette fois le traitement vas se faire pour les 3 canons c'est a dire qu'il y aura un calcule de médiane pour chaque canon et a la fin tout sera additionner pour avoir une image médiane couleur.

4.3.3 En sortie :

voici quelque médiane de notre liste comme sortie :

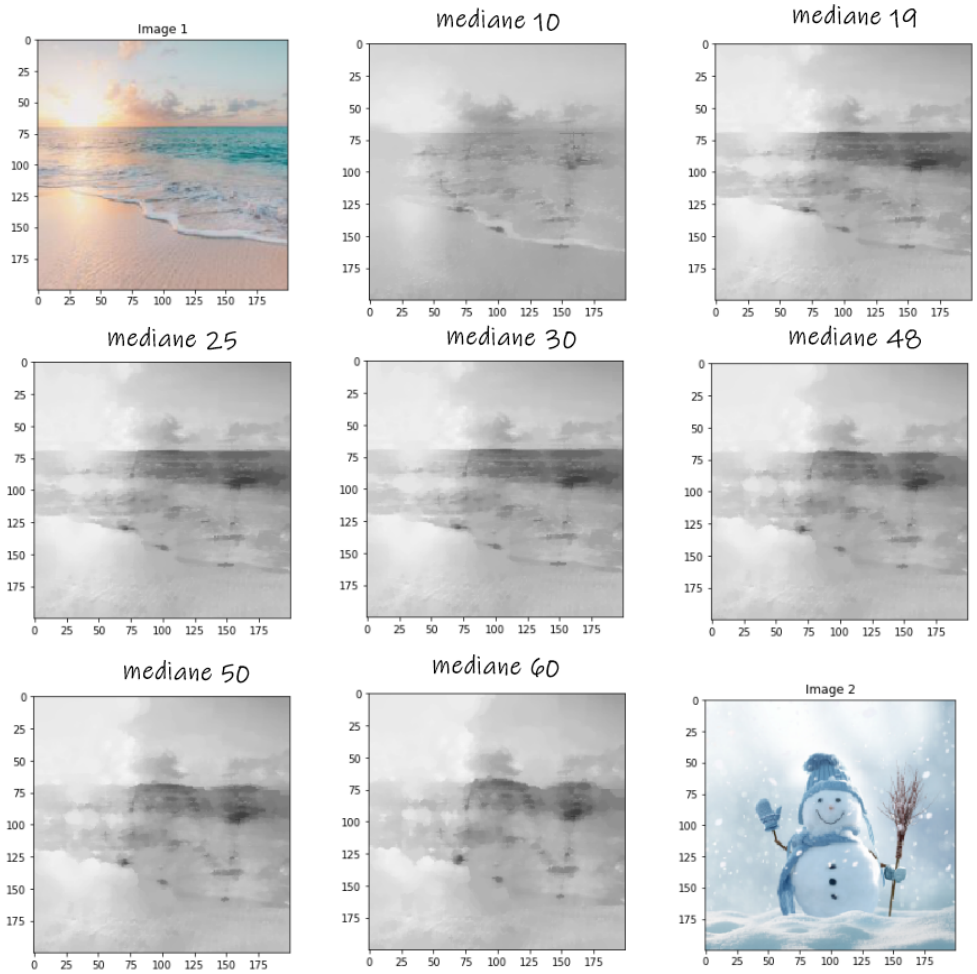


Figure 4.3: Images médiane couleurs

4.4 Remarque :

pour tout le traitement de nos trois type d'image on a fixer le paramètre n =6 , par conséquent le nombre de médianes est de : $(2^6) + 1 = 66$

(attention) dans le cas de l'image en couleur le nombres de médiane est de 66*3.

le temps de calcule de chaque médiane et de 5s. [djouad iskander et BATTACHE Nassim]

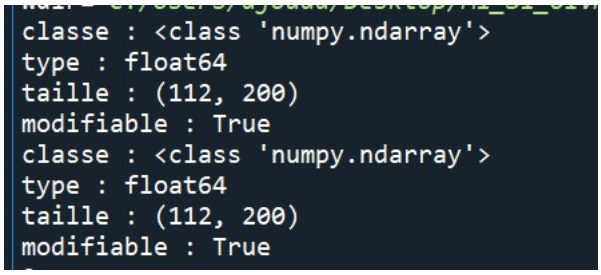


Figure 4.4: Images médiane Binaire

Conclusion

Dans ce projet on a effectuer une approche géométrique sur deux image de même type (binaire , gris ,couleur) , a fin d'estimer le flou optique .

on est parvenu a identifier les parties mobiles dans chaque image , l'aide de segmenta ion et la translation (regionproprs). calculer les déplacement apparent des points de l'image en calculons l'image médiane entre deux images afin d'estimer ou prédire les déplacements des formes .

pour reconstruire le mouvement dimensionnel à partir du mouvement dans le plan de l'image Calculez le flux optique et utilisez ses propriétés géométriques pour déduire des informations dimensionnelles sur la scène et le mouvement.

on a plus conclure que la forme médiane permet de suivre le changement d'une forme à une autre(Détection des changements temporels) .

Perspective

Définir une fonction qui dimensionne les inputs sans changer ou crée une nouvelle forme pour que la transformation soit le plus naturelle possible

réaliser une visualisation avec des images de type RGB avec des images de type RGBA .

insérer plus de médiane a la fois car notre algorithme crée 2 puis 4 puis 9 puis 17 .... 2(n-1)+1 médiane par itération .

optimiser le temps d'exécution.

Bibliography

[1] https://homepages.inf.ed.ac.uk/rbf/cvonline/local$_c$*opies*/*owens*/*lect*12/*node*4.*html*.

[2] https://scikit-image.org/.

[3] Morphologie mathématique IMAGES et MOUVEMENTS. Corrine lagorre.