Anamorphoses Cylindriques

Préambule :

L'image ci-dessus est une oeuvre de l'artiste hongrois Istvàn Orosz intitulée *A Magically Appearing Portrait of Jules Verne on the Mysterious Island* (1983).

À première vue il s'agit simplement d'explorateurs atteignant un navire échoué au milieu des glaciers, mais regardez mieux :

Une fois le cylindre réfléchissant posé, comme son nom le suggère, l'image prend un tout autre sens : le paysage s'y reflète et fait apparaître le visage de Jules Verne.

Un tel procédé est appelé une anamorphose cylindrique, notre objectif ici sera d'obtenir un résultat similaire : à partir d'une image initiale I_1 donnée de dimensions $H_1 \times L_1$, créer une image finale à imprimer I_2 de dimensions $H_2 \times L_2$, afin que s'y reflète l'image initiale.

Pour cela commençons par décider des dimensions de notre image finale, et créer une image vide que l'on cherchera donc à "colorier" :

In [1]:

```
from numpy import zeros, uint8
L_2, H_2 = 1280, 720
I_2 = zeros((H_2, L_2, 3), dtype = uint8)
```

Ensuite, il nous faut décider du rayon du cylindre que l'on souhaite utiliser pour notre modilisation. Pour cela nous avons tout simplement mesuré celui que nous avions à disposition et avons constaté qu'il correspondait à environ $13,4\%$ de la longueur d'une feuille $A4$.

In [2]:

```
from math import sqrt as racine
r =  .134*H_2
```

Puis il nous faut charger notre image initiale I_1, ici une photo du lycée Robert Doisneau, dans laquelle nous << piocherons >> les couleurs qui serviront à remplir notre image finale.

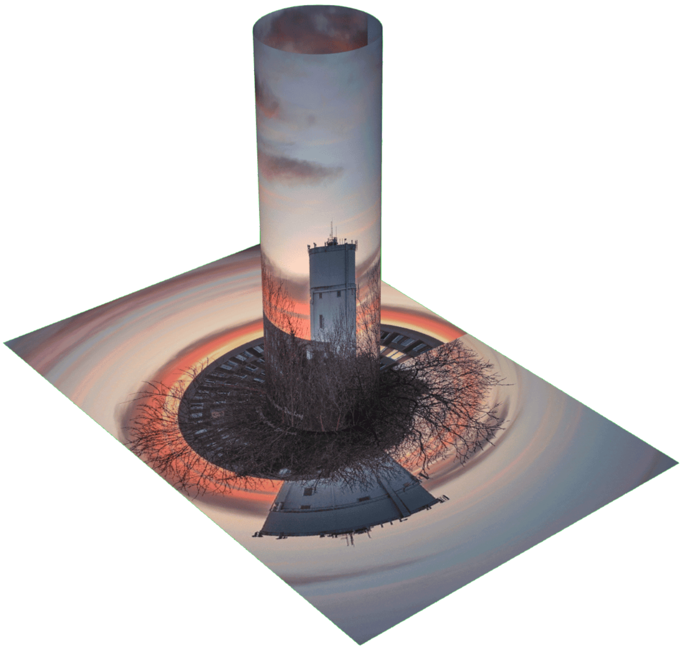
In [3]:

```
from PIL.Image import open

I_1 = open('Images/lycee.jpg')
L_1, H_1 = I_1.size
```

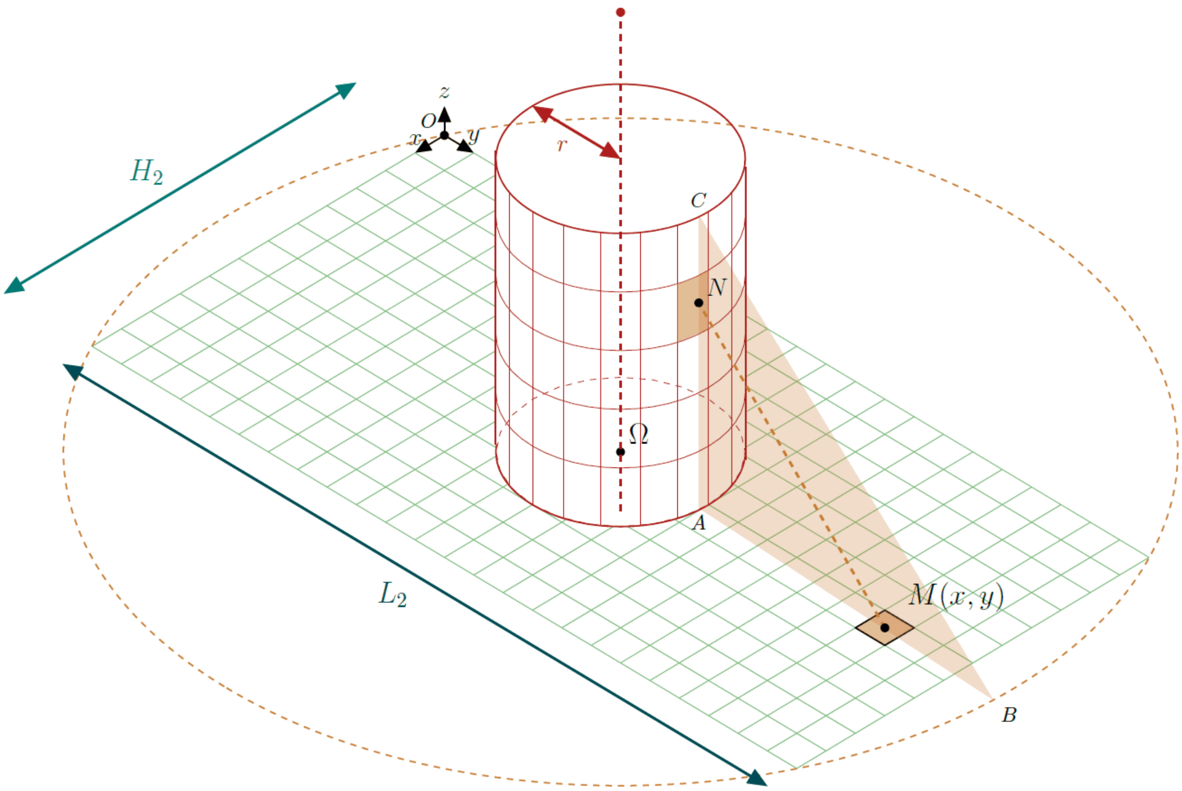
I. Projection

Dans un premier temps, il nous faut faire correspondre l'image finale << à plat >> avec la surface du cylindre sur laquelle elle est censée être réfléchie. L'idée étant d'obtenir un résultat similaire à l'image suivante :



On assimilera ainsi par << enroulement >> la hauteur du cylindre à la hauteur H_1 de l'image initiale.

Afin de détailler le processus permettant d'obtenir le résultat voulu, nous nous appuierons sur le schéma suivant :



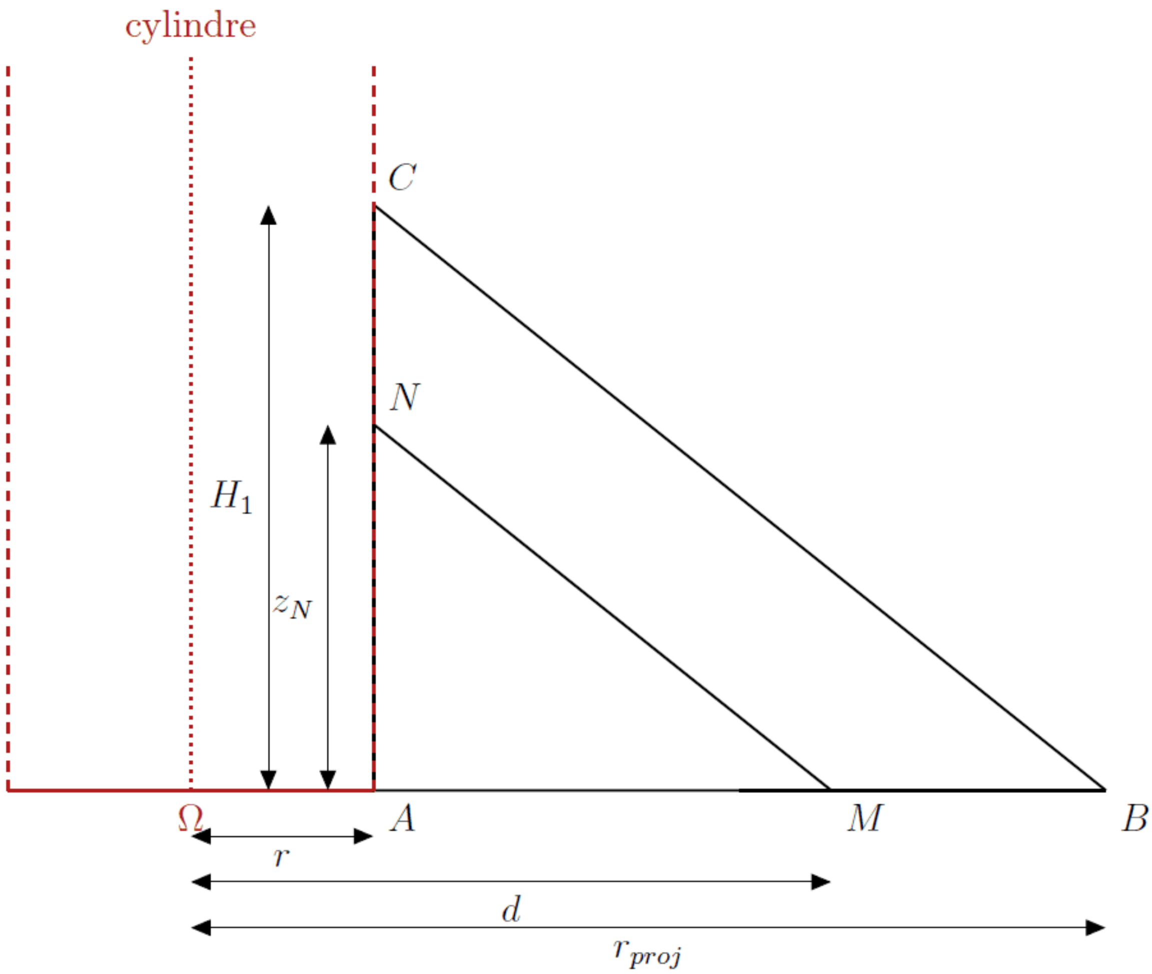
Nous avons donc :

● Une grille vide correspondant à l'image que l'on souhaite générer I_2, plongée dans l'espace munie du repère $\mathscr{R}_f = (O; \vec{u}; \vec{v}; \vec{w})$, dont la restriction $(O; \vec{u}; \vec{v})$ correspond aux coordonnées classiques d'un pixel sur une image numérique.

• Un cylindre fictif de rayon r placé au centre de I_2 en $\Omega\left(\frac{L_2}{2};\frac{H_2}{2};0\right)$ dans $\mathscr{R}_f$ auquel on associe le repère $\mathscr{R}_{cyl}$ correspondant aux coordonnées cylindriques.

Ainsi pour un pixel de coordonnées $M(x;y;0)$ dans $\mathscr{R}_f$ sur I_2, on considère les points A et B situés respectivement à l'intersection de la demi-droite $[\Omega M)$ et des cercles de centre Ω et de rayons respectifs r et ΩO.

De plus, on définit également le point C de coordonnées $(x_A;y_A;H_1)$, de sorte à associer au point M un point N à la surface du cyclindre déterminé comme étant l'intersection entre la droite (AC) et la parallèle à (BC) passant par M.



En se plaçant dans le plan (ABC) représenté ci-dessus, on obtient en appliquant le théorème de Thalès : $\frac{z_N}{H_1}=\frac{d-r}{r_{proj}-r}$ en notant $d=\Omega M$.

La hauteur du point N recherché est donc $z_N=H_1\times\frac{d-r}{r_{proj}-r}$.

Nous pourrons donc la calculer en définissant la fonction *hauteur* définie ci-après :

In [4]:
```
def hauteur(d) :
    return H_1*(d - r)/(r_proj - r)
```

Pour que cette dernière soit effective, il nous faut pouvoir calculer les différentes distances entrant en jeu, ce que nous ferons à l'aide de la fonction suivante :

In [5]:

```
from math import sqrt as racine
def distance(x, y) :
    return racine((x - L_2//2)**2 + (y - H_2//2)**2)
```

La fonction distance permet en particulier de calculer la distance $r_{proj} = \Omega O$:

In [6]:

```
r_proj = distance(0,0)
```

Le point N ainsi associé est donc situé sur un cercle de rayon r et de centre $\left(\frac{L_2}{2}; \frac{H_2}{2}; z_N\right)$ dans le plan $z = z_N$.

D'où : $\begin{cases} x_N = r\cos(\theta) + x_\Omega = r\cos(\theta) + \dfrac{L_2}{2} \\ y_N = r\sin(\theta) + y_\Omega = r\sin(\theta) + \dfrac{H_2}{2} \end{cases}$

Le point N a donc pour coordonnées cylindriques $(r; \theta; z_N)$ où θ est une mesure de l'angle orienté $(Ox; \overrightarrow{\Omega M})$.

Pour déterminer une valeur convenable de θ, nous utiliserons la fonction prédéfinie atan2 prenant en argument l'ordonnée et l'abscisse (dans cet ordre) du vecteur $\overrightarrow{\Omega M}$ et renvoyant la mesure principale de l'angle orienté $(\vec{u}, \overrightarrow{\Omega M})$.

In [7]:

```
from math import atan2, pi

def angle(x, y) :
    return atan2(y - H_2/2, x - L_2/2) + pi/2
```

Pour des raisons esthétiques il peut être intéressant de "décaler" l'angle obtenu, ce qui reviendrait matériellement à faire tourner l'image I_1 sur le cyclindre et donc sa projection I_2. Nous avons ici effectué un déphasage de $\frac{\pi}{2}$ (un quart de tour) afin de rendre le château d'eau entièrement visible sur la projection.

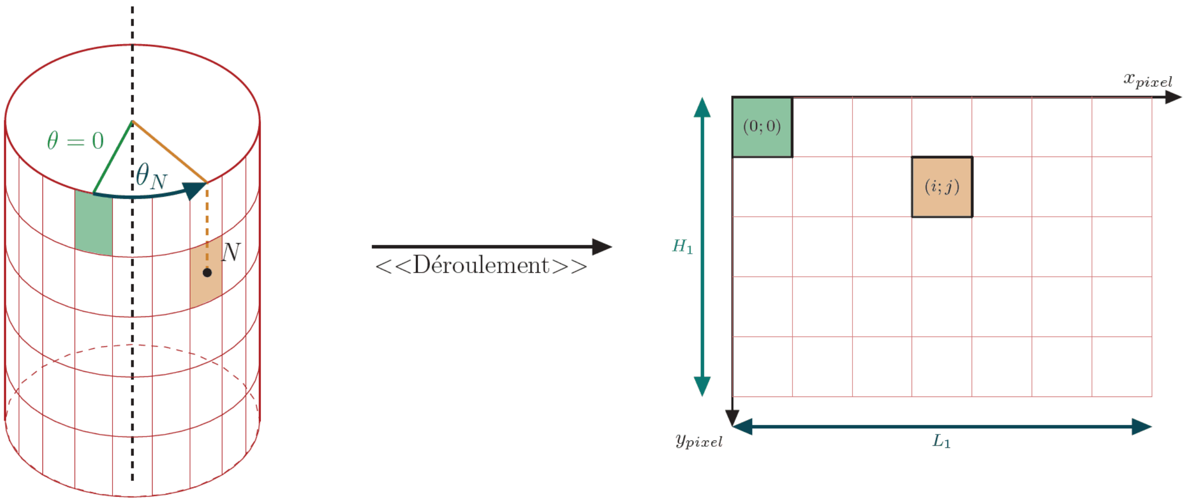
II. Déroulement

Puisque nous avons réussi à expliciter la correspondance entre l'image finale et le cylindre, il nous faut à présent faire de même avec l'image initiale, pour cela il suffit d'imaginer qu'elle est $<<$ enroulée $>>$ autour du cylindre, comme représenté ci-après :

Enroulement de l'image initiale

Autrement dit, une fois les coordonnées z_N et θ obtenues grâce à la première partie, il reste à les associer aux coordonnées $(i; j)$ de l'image numérique initiale I_1 obtenues par << déroulement >>, afin d'en extraire sa couleur.

Pour cela nous nous appuierons sur le schéma suivant :



• la coordonnée j est liée à la hauteur de N, z_N, elle correspond à la distance entre N et le point de coordonnées cylindriques (θ, H_1), ainsi : $j = H_1 - z_N$.

• la coordonnée i est liée à l'angle θ par proportionalité : $i = \dfrac{L_1 \times \theta}{2\pi}$.

D'où la fonction suivante, qui va nous aider à attribuer à chaque pixel vide sa couleur :

In [8]:

```
def couleur(z, theta) :
    j = int(H_1 - z)
    i = int( L_1 * theta /(2*pi))
    return i, j
```

Reste à appliquer l'ensemble des raisonnements précédents, au travers des différentes fonctions définies, à chaque pixel (x, y) de l'image finale à l'aide d'une double boucle **for** :

In [9]:

```
for x in range(L_2) :
    for y in range(H_2) :
        # on calcule la distance entre Omega et M
        d = distance(x, y)
        # si M est dans le disque de projection, on cherche la couleur associee
        if  d >= r :
            theta = angle(x, y)
            z = hauteur(d)
            i, j = couleur(z, theta)
            I_2[y,x] = I_1.getpixel((i,j))
        else : # sinon on colorie le pixel en blanc
            I_2[y,x] = (255,255,255)
```

Puis à afficher l'image ainsi obtenue :

In [10]:

```
from PIL.Image import fromarray
I_2 = fromarray(I_2)
I_2.show()
```

Et enfin à l'enregistrer afin de l'imprimer et observer le résultat final :

In [11]:

```
I_2.save('lycee_projection.jpg')
```