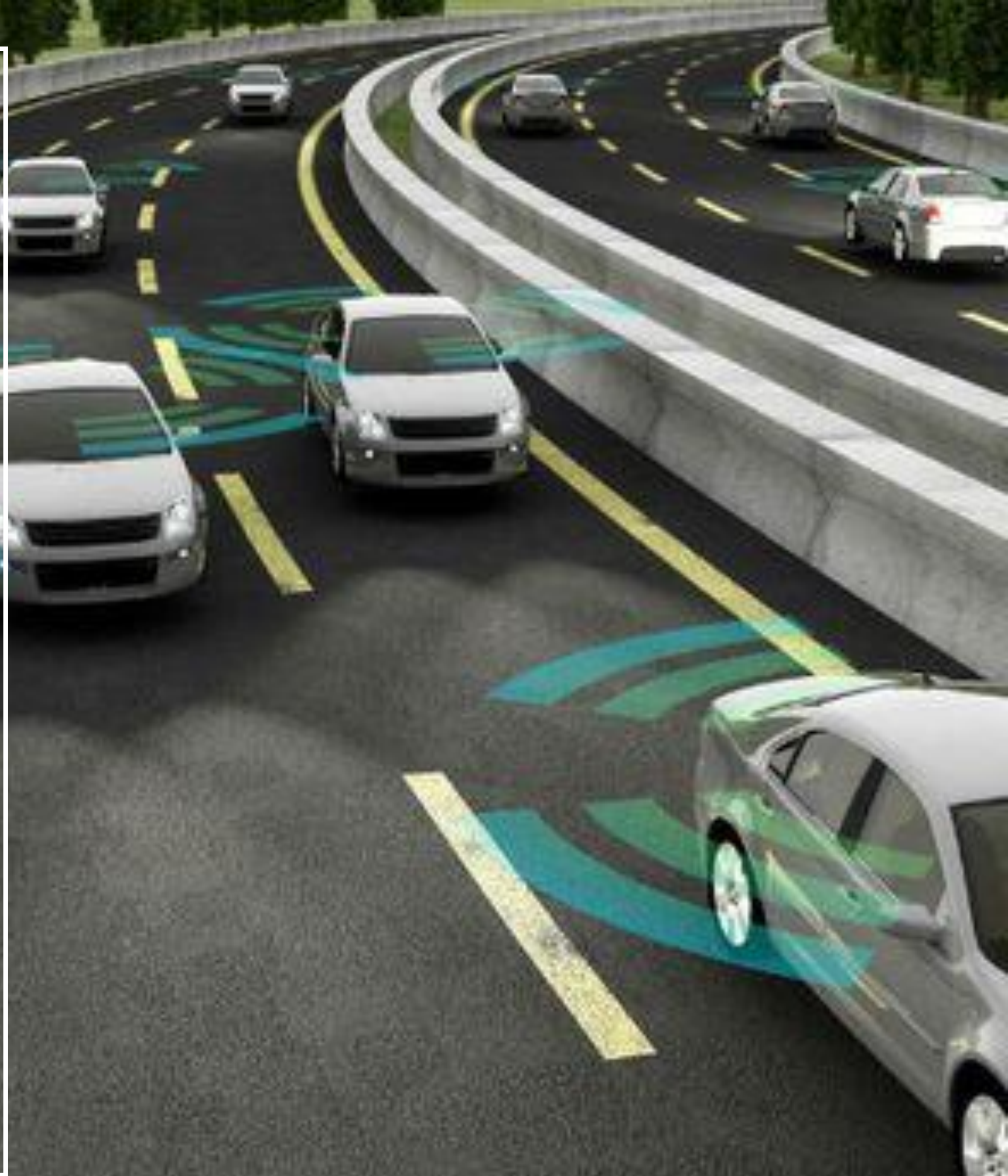
TRANSPORT D'ÉNERGIE LUMINEUSE EN MILIEU INHOMOGÈNE

APPLICATION AU LIDAR

Adrien Salem-Sermanet

Session 2019

32273

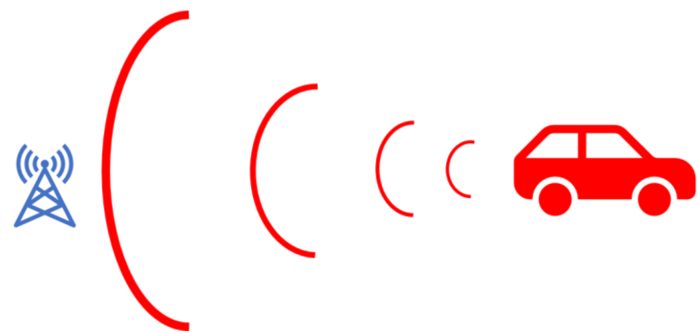


Fonctionnement général du LiDAR

Type étudié : Télémètre laser à balayage

Modèle tridimensionnel de son environnement : $z = \frac{c}{2}(t_{écho,max} - t_0)$

Etape 1 : Emission d'une impulsion laser



Etape 2 : Réception de l'écho réfléchi par la cible

Problématique et plan

Dans quelle mesure les conditions météorologiques impactent-elles la précision du LiDAR ?

I. Perte de puissance - Approche continue

II. Perte de puissance - Approche discrète

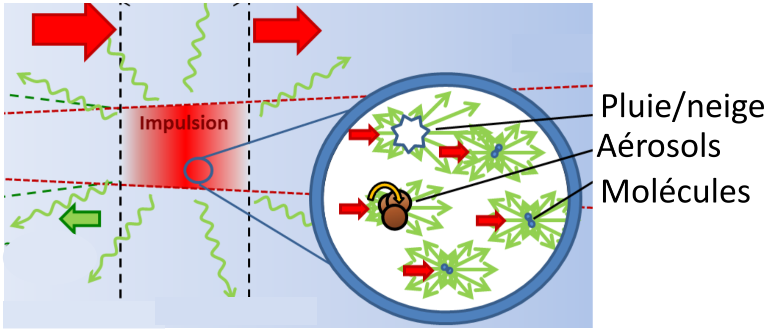
III. Déviation : hautes températures et pluie

I. Perte de puissance
Approche continue

A. Perte de puissance dans l'atmosphère

Deux causes :

1. Par **absorption** :
   coefficient d'absorption α(z,λ) en m^{-1}
2. Par **diffusion** :
   coefficient de diffusion β(z,λ) en m^{-1}

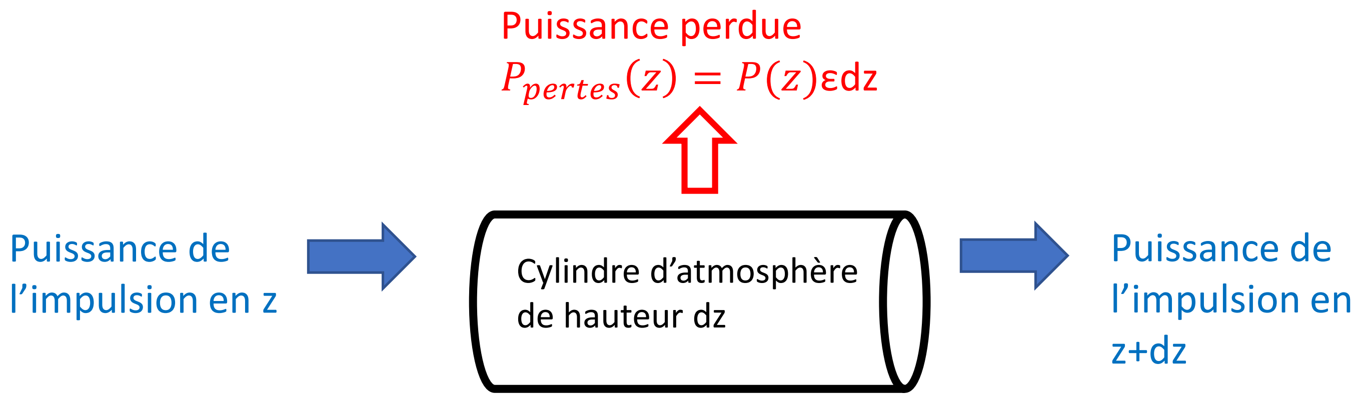


On note ainsi ε(z,λ) : coefficient d'extinction avec ε(z,λ) = α(z,λ) + β(z,λ) en m^{-1}

Premier modèle : Atmosphère = milieu d'atténuation uniforme

ε(z,λ) = ε constant à λ fixée

A. Perte de puissance dans l’atmosphère



Loi de Beer-Lambert :

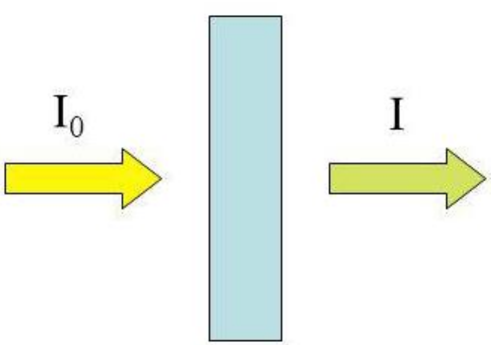
Bilan de puissance pour une impulsion laser entre z et z+dz :

P(z+dz)=P(z)(1- εdz)

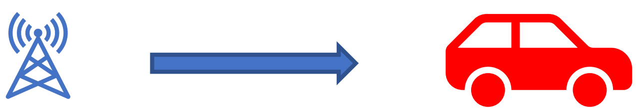
Donc **P(z)=$\boldsymbol{P_0}$ T(z)** avec la transmittance T(z) $= \frac{P(z)}{P_0} = e^{\int_0^z -\varepsilon dz'} = e^{-\varepsilon z}$

Analogie avec la spectrophotométrie :

Absorbance A = -log($\frac{I}{I_0}$) = -log(T) avec T= $\frac{I}{I_0}$



B. Equation LiDAR

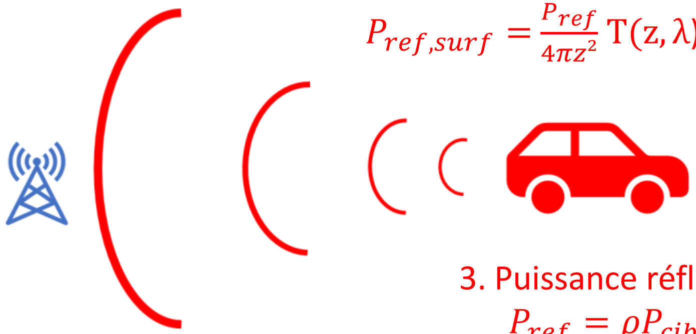


1. Puissance émise

$P_{émise} = P_0 \, \tau_T$

2. Puissance reçue par la cible

$P_{cible} = P_0 \, \tau_T \, \mathrm{T}(z, \lambda)$



4. Puissance rétrodiffusée surfacique

$P_{ref,surf} = \frac{P_{ref}}{4\pi z^2} \mathrm{T}(z, \lambda)$

3. Puissance réfléchie

$P_{ref} = \rho P_{cible}$

5. Puissance captée par le LiDAR

$P_{LiDAR} = A_r \tau_R P_{ref,surf}$

B. Equation LiDAR

$$P_{LiDAR}(z) = \frac{P_0 A_r \tau_T \tau_R \rho}{4\pi z^2} e^{-2\varepsilon z}$$

avec : A_r la surface de réception du LiDAR (m²)

τ_T le coefficient d'efficacité de transmission (entre 0 et 1 sans unité)

ρ l'albedo de la cible (entre 0 et 1 sans unité)

τ_R le coefficient d'efficacité de réception (entre 0 et 1 sans unité)

C. Simulation numérique

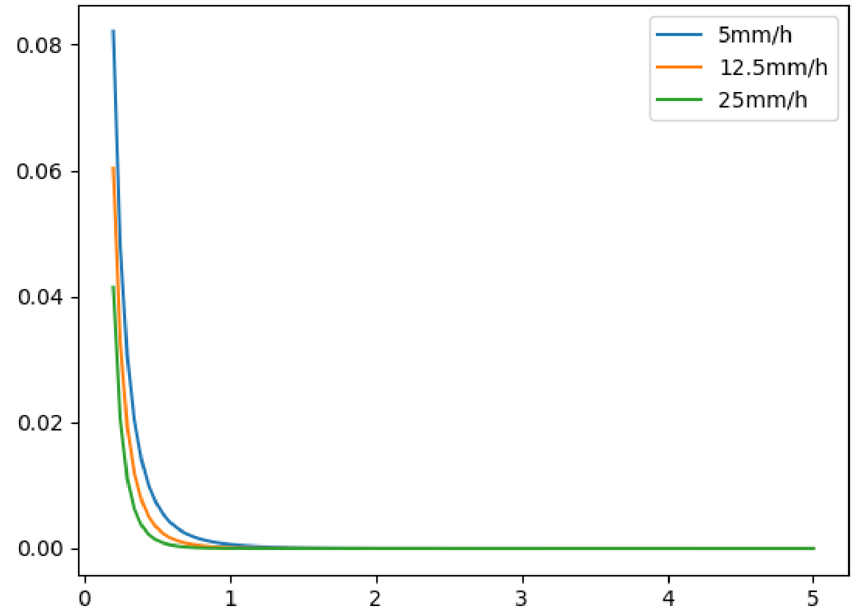
Valeur empirique pour ε :
ε =$0.02R^{0.6}$
avec R : précipitation en mm/h

Equation LiDAR simplifiée :

$$\boxed{P_r = \frac{K}{z^2} e^{-2\times 0.02R^{0.6}z}}$$

K en Wm^2

Puissance relative reçue (sans unité)

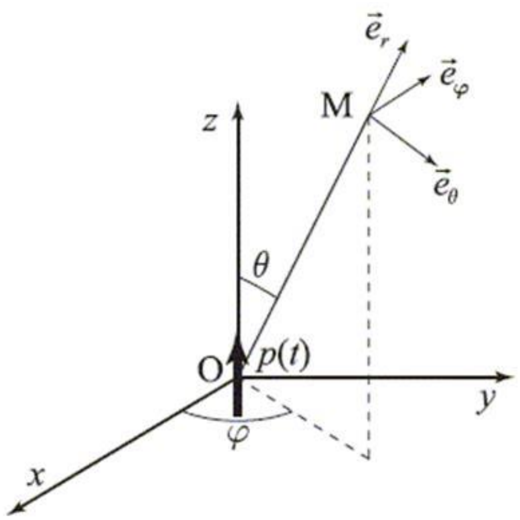


Distance à la cible (m)

Puissance relative reçue en fonction de la distance à la cible pour plusieurs valeurs de R

II. Perte de puissance
Approche discrète

A. Modélisation



Faisceau : Onde électromagnétique $(\overrightarrow{E}, \vec{B})$

Goutte : Rayon b

Moment dipolaire induit par $\vec{E}$: $\vec{p} = p_0 \vec{E}$ avec $p_0 = 4\pi\varepsilon_0 \frac{\varepsilon-1}{\varepsilon+2} b^3$

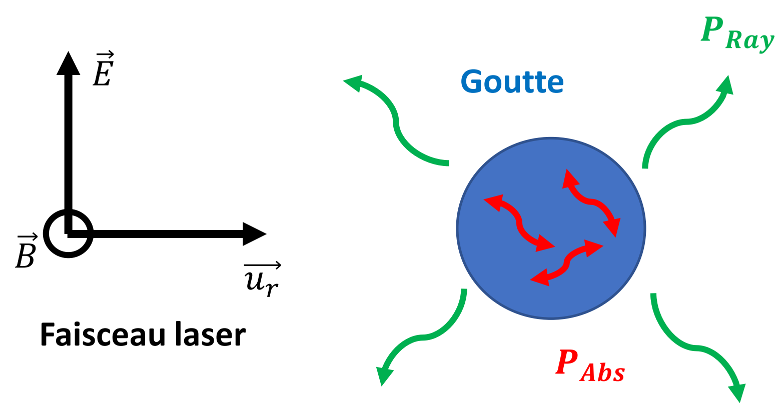
Rayonnement dipolaire : $p_{Ret}(t) = p(t - {}^{r}/_{c})$

$\vec{E}_{Ray}$ $selon$ $\overrightarrow{u_\theta}$ et $\vec{B}_{Ray}$ $selon$ $\overrightarrow{u_\varphi}$

B. Absorption et diffusion d’énergie lumineuse

Théorème de Pointing appliqué à la goutte en volumique :

$$\underbrace{\frac{\partial w}{\partial t}}_{\text{Variation de puissance volumique}} = -\underbrace{\mathrm{div}(\vec{\Pi})}_{\text{Puissance volumique rayonnée}} - \underbrace{\vec{j}\cdot\vec{E}}_{\text{Puissance volumique absorbée}}$$



B. Absorption et diffusion d'énergie lumineuse

Puissance absorbée :

$< P_{Abs} > = \iiint_{V_{goutte}} < \vec{j} \cdot \vec{E} > . \overrightarrow{d\tau} = < \vec{j} \cdot \vec{E} > V_{goutte}$ avec $\vec{E}$ uniforme ($\lambda = \frac{c}{f_0}$>>b)

$\vec{j} = \frac{d\vec{p}}{dt}$ = -iω $\vec{p}$

Donc $\boxed{< P_{Abs} > \sim b^6 f_0}$

$V_{goutte} = \frac{4}{3}\pi b^3$

Green-Ostrogradski

Puissance rayonnée :

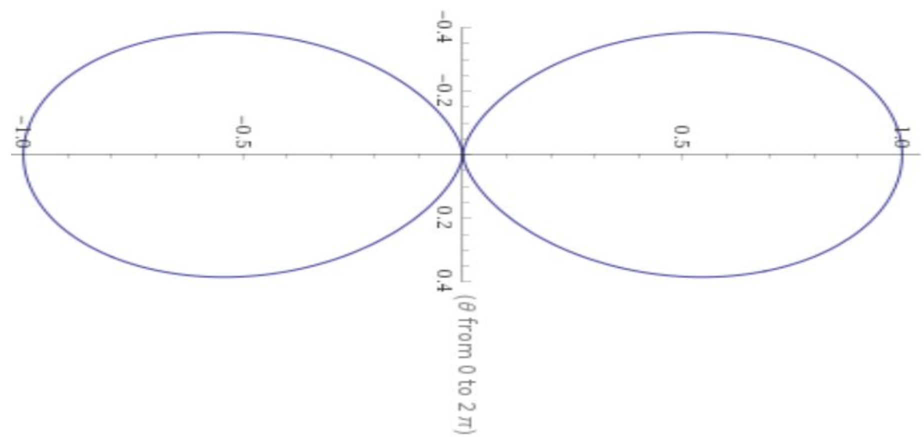
$< P_{Ray} > = \iiint_{V_{goutte}} div(< \vec{\Pi}(r,\theta) >) d\tau = \oiint_{S_{goutte}} < \vec{\Pi}(r,\theta) > . \overrightarrow{ds}$

Donc $\boxed{< P_{Ray} > \sim b^6 f_0}$

B. Absorption et diffusion d’énergie lumineuse

Puissance moyenne rayonnée par unité de surface :

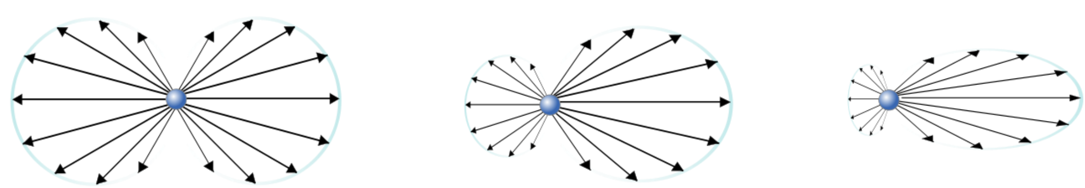
$$< \vec{\Pi}(r,\theta) >= \frac{1}{\mu_0} < \vec{E} \wedge \vec{B} > = \mathrm{A}\, \frac{(\sin\theta)^2}{r^2}\, \overrightarrow{u_r}$$



Directivité du rayonnement :

On trace R= f(θ)= $\frac{<\vec{\Pi}(r,\theta)>}{<\overrightarrow{\Pi_{Max}}(r)>}$ = $(\sin\theta)^2$

en coordonnées polaires



C. Modèle corpusculaire probabiliste

International Journal of Antennas and Propagation

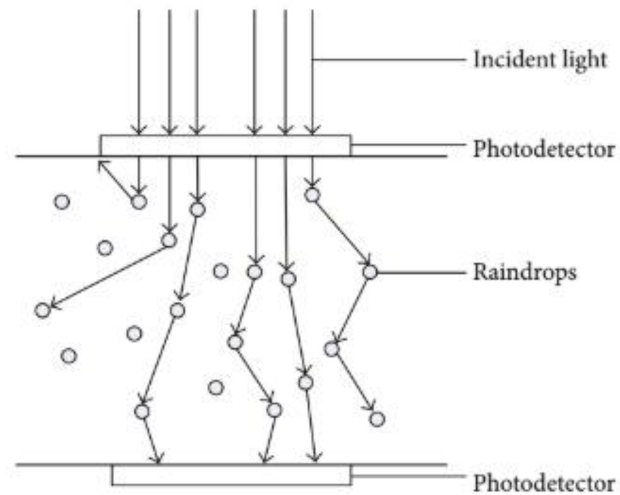


FIGURE 1: Schematic diagram for photons propagation model in stochastic rainfall.

Principe :

- Faisceau lumineux = flux de photons
- Gouttes et déviation générées pseudo-aléatoirement

Paramètres :

- R intensité de la pluie en mm/h
- ω pulsation du photon (E= $\hbar\omega$) en rad/s
- μ_s et μ_a coefficients de diffusion et d'absorption dépendant de R en m^{-1}
- L distance entre les gouttes en m
- Paramètre d'anisotropie g dans [-1;1] sans unité

C. Modèle corpusculaire probabiliste

A chaque goutte :

- ω <-- ω $(1 - \frac{\mu_s}{\mu_s+\mu_a})$
- Φ angle d'azimuth aléatoire
- Θ angle de colatitude = H(g, ξ) fonction de phase de Henyey-Greenstein
- L = $\frac{-\ln(\xi)}{\mu_s+\mu_a}$ avec ξ aléatoire dans]0;1[

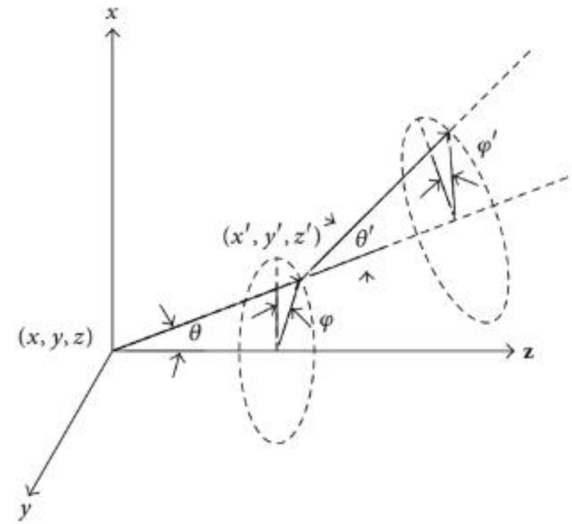
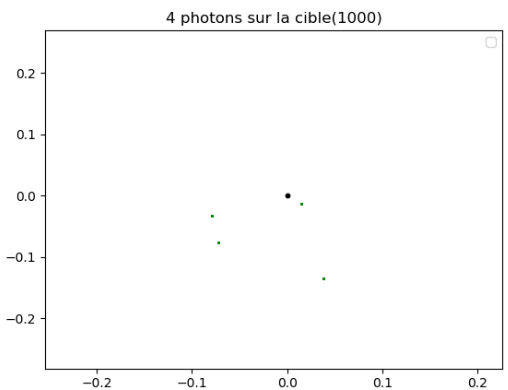
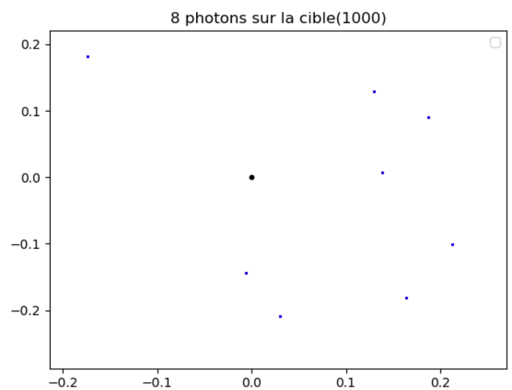


FIGURE 2: Coordinates of photon propagation.

C. Modèle corpusculaire probabiliste

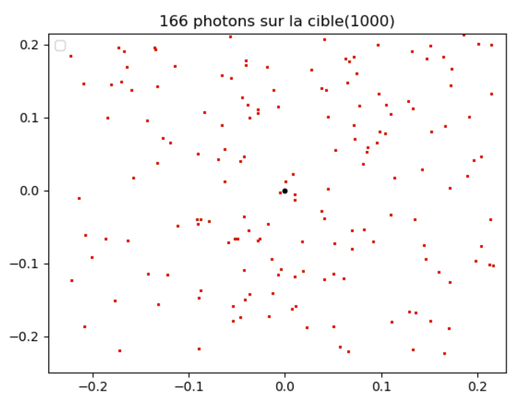


g=0.9

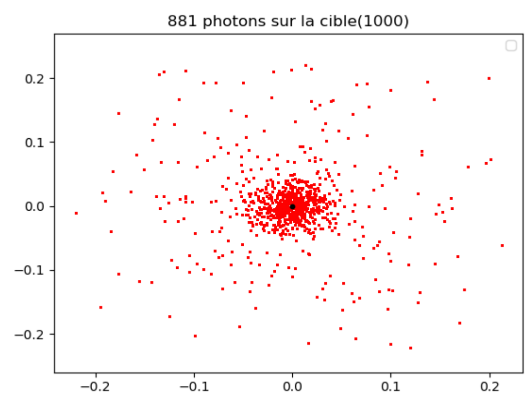


g=0.95

Photons sur l'écran de réception du LiDAR (0.2x0.2 m²) pour une cible de 2x2m² à 10m du LiDAR et R=25mm/h



g=0.99

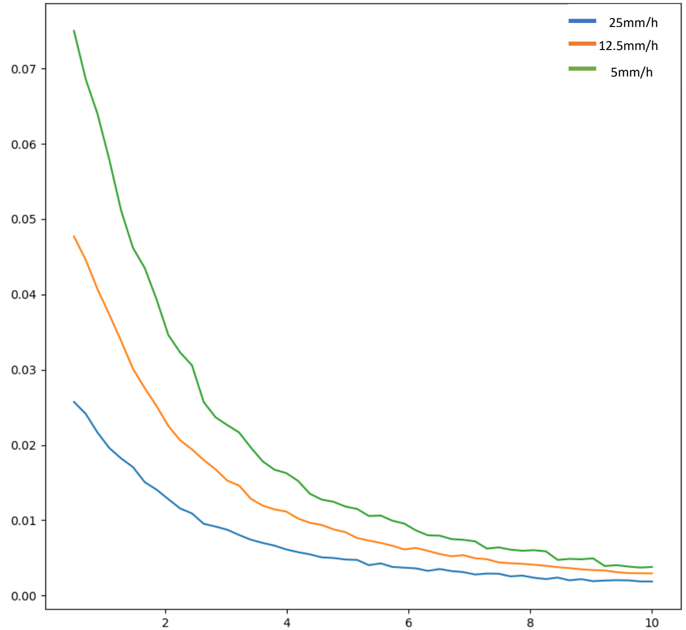


g=0.999

Code couleur énergie :
Vert < Bleu < Rouge

C. Modèle corpusculaire probabiliste

Puissance relative reçue (sans unité)



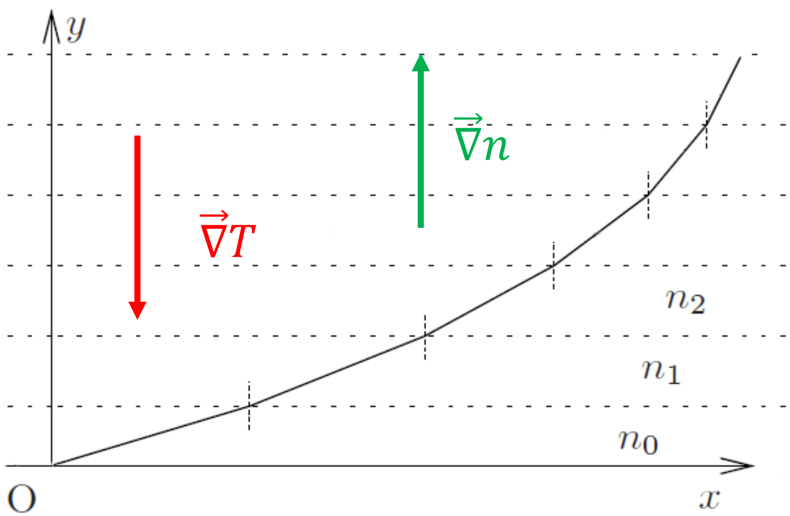
Distance à la cible (m)

Puissance relative reçue pour une cible de 2x2 m^2 et g=0.99 pour différentes valeurs de précipitations

III. Déviation

1. Hautes températures

A. Equation de propagation théorique



Modèle : Milieu stratifié et 3ᵉ loi de Descartes

Loi de Gladstone : (n-1)T=0.082K pour l'air

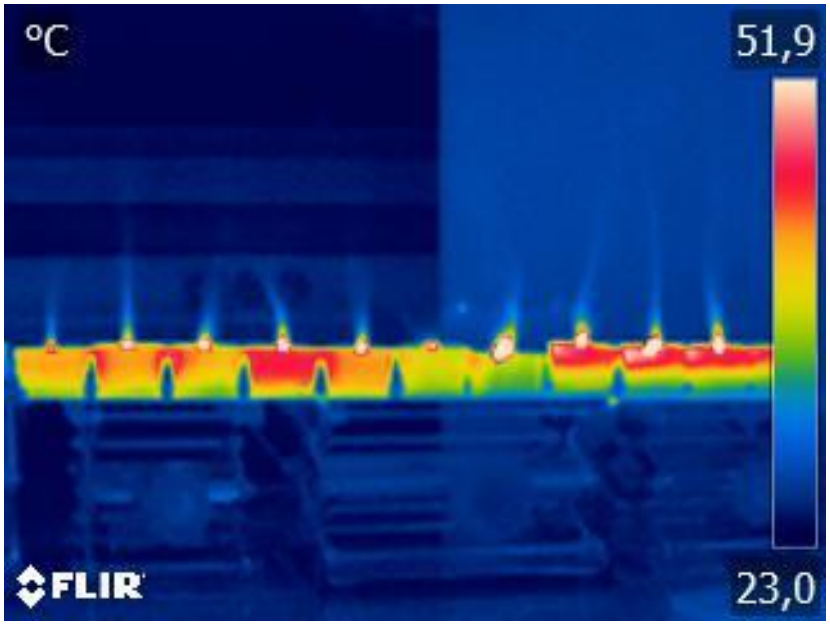
T(x) parabolique (interpolation)

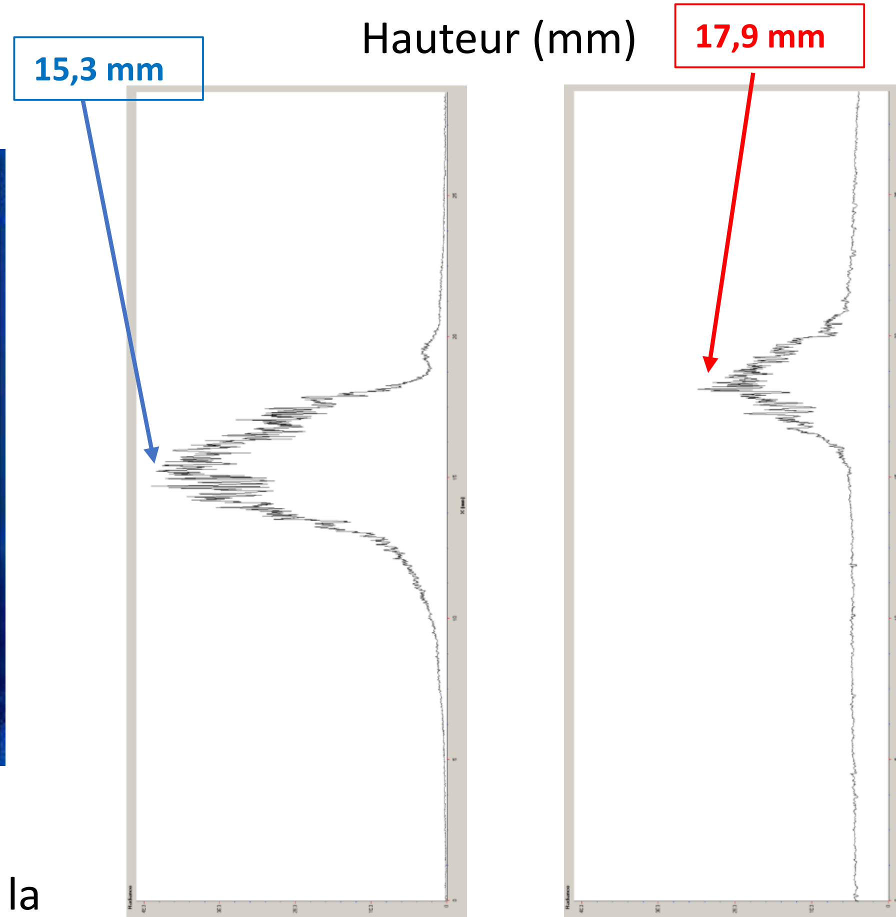
Equation différentielle de trajectoire:

$$\frac{d^2y}{dx^2} = -\frac{K}{C_0^2}\left(\frac{K}{T}+1\right)\frac{1}{T^2}\frac{dT}{dy}$$

B. Déviation expérimentale du faisceau

Longueur gradient : 1m
Distance laser/cible : 5.3 m



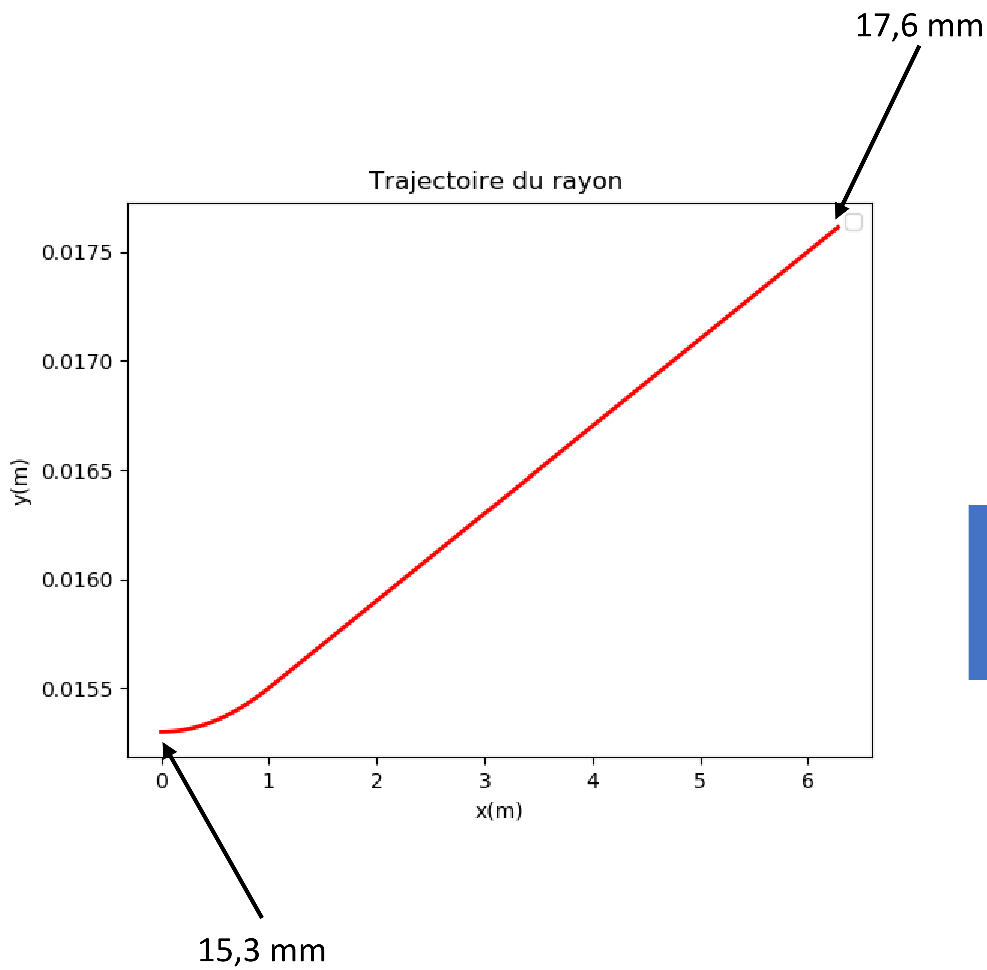


Radiance reçue par la caméra CCD ($W{\cdot}m^{-2}{\cdot}sr^{-1}$)

Sans gradient

Avec gradient

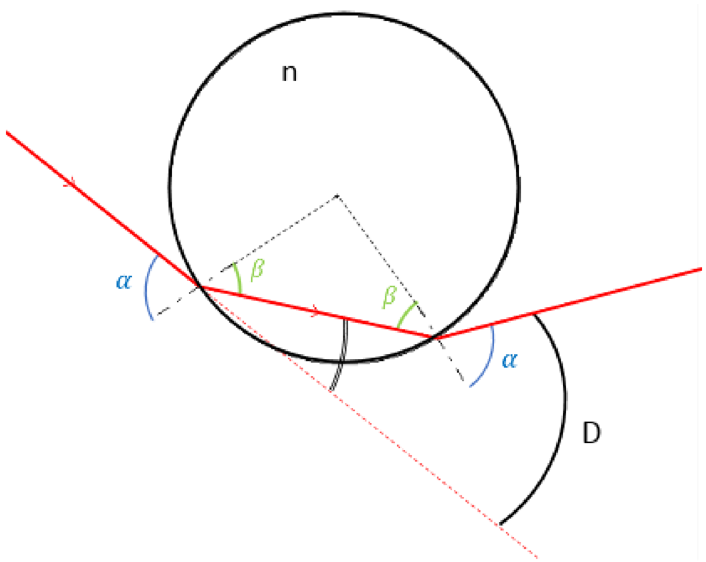
C. Simulation numérique et conclusion

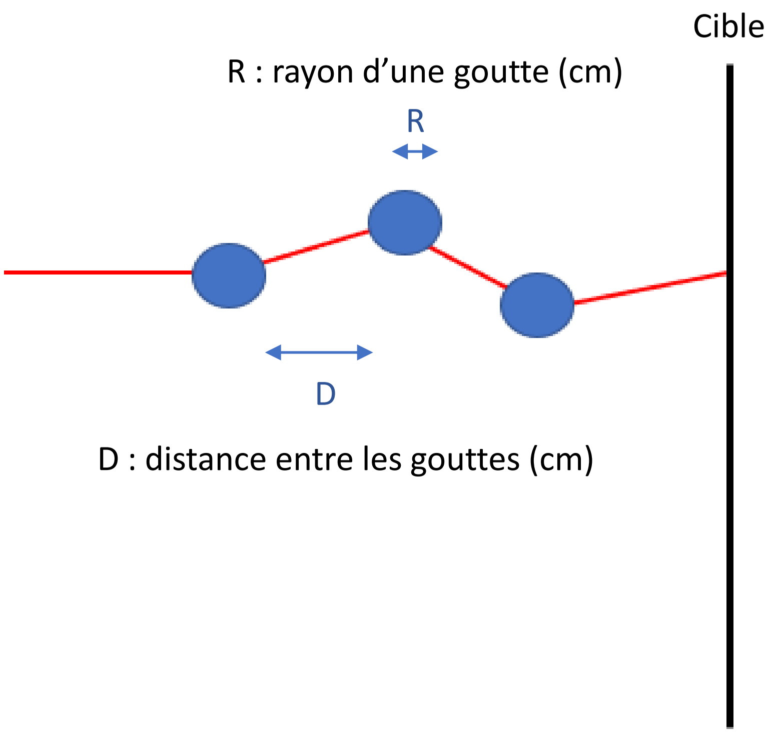


	Expérience	**Simulation**
Déviation finale	2.6 mm	2.3 mm

III. Déviation
2. Pluie

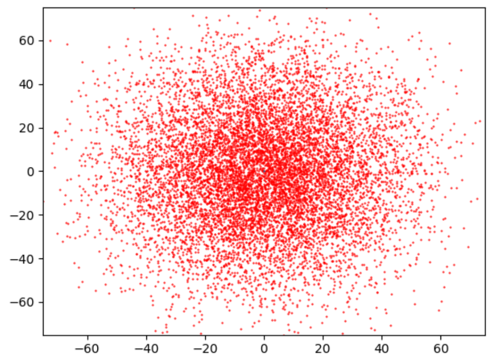
A. Déviation du rayon par une goutte d'eau





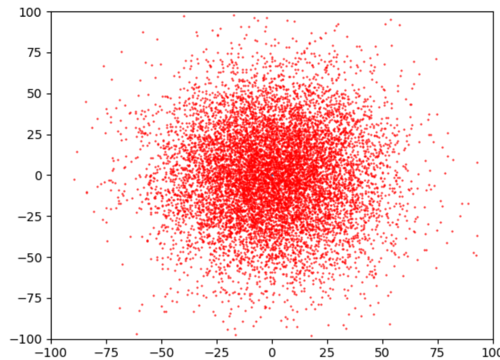
B. Simulation numérique (R fixé et D varie)

R=0.3   D=30 (cm)



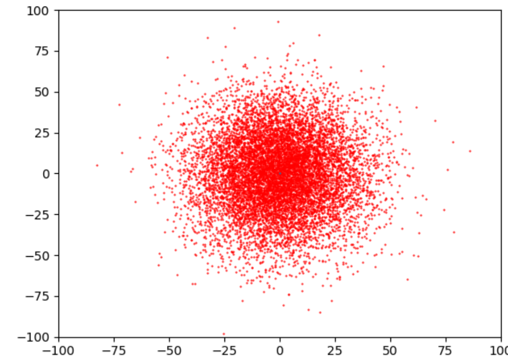
Vx=1109
Vy=1115

R=0.3   D=20 (cm)

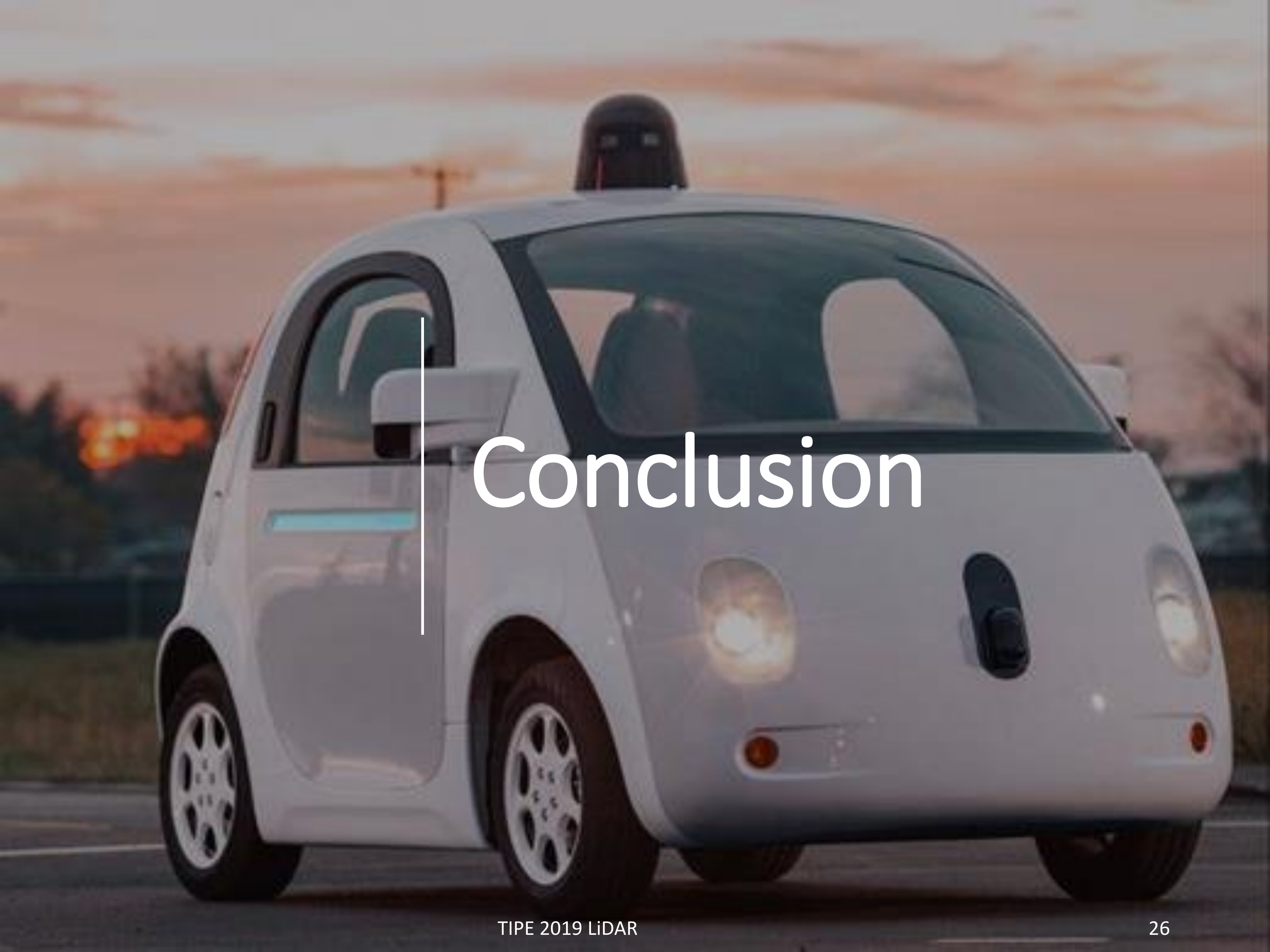


Vx=615
Vy=609

R=0.3   D=10 (cm)



Vx=363
Vy=369



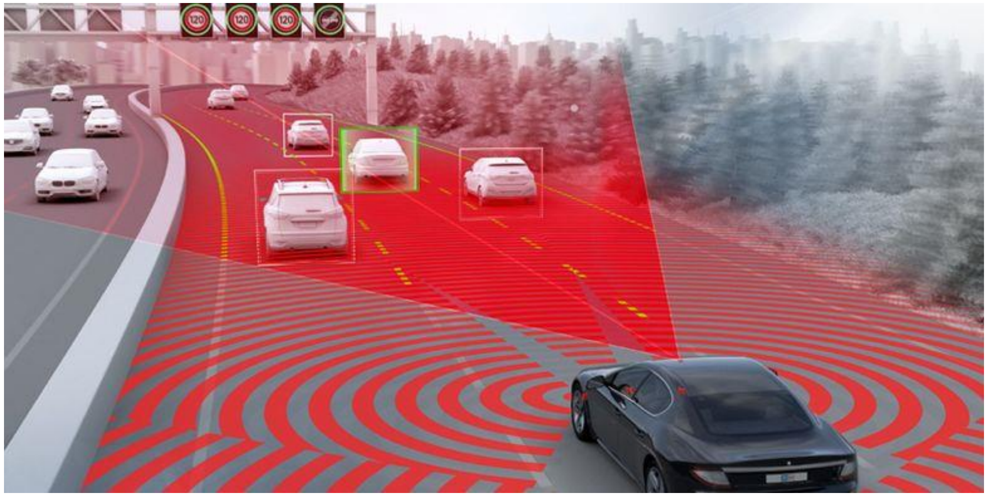
Conclusion

Conclusion

Impacts des conditions météorologiques sur les performances du LiDAR :

- Perte de puissance importantes
- Déviation par la pluie
- Peu d'effet du gradient thermique

Solutions ?



Annexes

Equation de propagation du faisceau

En appliquant la 3ème loi de Descartes on obtient la relation suivante :

$$n_i \cos(\alpha_i) = n_{i+1} \cos(\alpha_{i+1})$$

La quantité $C_0 = n(y)\cos\big(\alpha(y)\big)$ est alors constante.

En coordonnées cartésiennes : $\cos\big(\alpha(y)\big) = \frac{dx}{\sqrt{dx^2+dy^2}} = \frac{1}{\sqrt{1+(\frac{dy}{dx})^2}}$

D'où l'équation différentielle suivante :

$$n(y) = C_0\sqrt{1+\left(\frac{dy}{dx}\right)^2}$$

En dérivant selon x, on obtient :

$$\frac{d^2y}{dx^2} = \frac{1}{2}\frac{d}{dy}\left(\left(\frac{n}{C_0}\right)^2\right)$$

Loi de Gladstone appliquée à l'air : $(n-1)\times T = K = 0.082\ K$

Donc $\frac{dn^2}{dy} = 2n\frac{dn}{dy} = -2\left(\frac{K}{T}+1\right)\frac{K}{T^2}\frac{dT}{dY}$

$$\frac{d^2y}{dx^2} = -\frac{K}{C_0^2}\left(\frac{K}{T}+1\right)\frac{1}{T^2}\frac{dT}{dy}$$

Moment dipolaire d'une goutte d'eau

Matériau isotrope soumis à un champ électrique peu intense → Polarisation électrique et atomique/ionique

Polarisation microscopique : P=$\varepsilon_0\chi$E avec ε_0 permittivité du vide ($\varepsilon_0\chi$ = α_e la polarisabilité électrique)

χ susceptibilité électrique du matériau

Or, $\alpha_e = \frac{3M\varepsilon_0}{N_A d}\frac{n^2-1}{n^2+2}$ avec n indice optique du milieu ($n^2=\frac{\varepsilon}{\varepsilon_0}$) et d sa densité

De plus, P=$\frac{dp}{dV}$ donc le moment dipolaire total p = $\iiint P\,dV$

Ainsi, pour une goutte d'eau :

$$p = \iiint P\,dV \approx \frac{3M\varepsilon_0}{N_A d}\frac{n^2-1}{n^2+2} V_{goutte} E = \frac{3M\varepsilon_0}{N_A d}\frac{n^2-1}{n^2+2}\frac{4}{3}\pi b^3 E \sim 4\pi\varepsilon_0 \frac{\varepsilon-1}{\varepsilon+2} b^3\ E$$

Modélisation

Faisceau : Onde électromagnétique $(\vec{E}, \vec{B})$ localement plane dans le vide $\vec{E} = E_0 e^{i\omega(t-\frac{r}{c})}\overrightarrow{u_z}$ et $\vec{B} = \frac{\overrightarrow{u_z} \wedge \vec{E}}{c}$ (relation de structure)

Vecteur de Pointing : $\vec{\Pi} = \frac{1}{\mu_0} \vec{E} \wedge \vec{B}$

Goutte : Rayon b

Moment dipolaire induit par $\vec{E}$: $\vec{p} = p_0\, \vec{E}$ avec $p_0 = 4\pi\varepsilon_0 \frac{\varepsilon-1}{\varepsilon+2} b^3$

Rayonnement dipolaire : $p_{Ret}(t) = p(t - {}^{r}/{}_{c})$

$\vec{E}_{Ray} = \frac{\mu_0 \sin(\theta)}{4\pi r} \ddot{p}_{Ret}(t)^2 \overrightarrow{u_\theta}$ et $\vec{B}_{Ray} = \frac{\mu_0 \sin(\theta)}{4\pi r c} \ddot{p}_{Ret}(t)^2 \overrightarrow{u_\theta}$

Calculs de puissances

Puissance rayonnée par la goutte

$< P_{Ray} >= \int_{\theta=0}^{\pi} \int_{\varphi=0}^{2\pi} < \vec{\Pi}(r,\theta) >. \overrightarrow{dS}$ avec $\overrightarrow{dS} = r^2 \sin\theta d\theta\, d\varphi$

$< P_{Ray} >= \frac{\mu_0 p_0^2}{12\pi c} (2\pi f_0)^4$ donc $\sim b^6 f_0$

Puissance moyenne rayonnée par unité de surface

$< \vec{\Pi}(r,\theta) >= \frac{1}{\mu_0} < \vec{E} \wedge \vec{B} >$ = $\frac{\mu_0}{16\pi^2 cr} < \ddot{p}_{Ret}{}^2 > (\sin\theta)^2\, \overrightarrow{u_r}$

= $\frac{\mu_0 p_0^2 (2\pi f_0)^4}{32\pi^2 cr^2} (\sin\theta)^2 \overrightarrow{u_r}$ = A $\frac{(\sin\theta)^2}{r^2} \overrightarrow{u_r}$

Diffusion de Mie

Paramètre de taille x = $\frac{2\pi n a}{\lambda}$ avec : n indice optique

a rayon de la particule

λ longueur d'onde du faisceau

1. x<<1 : Diffusion de Rayleigh (couleur bleue du ciel en 1/lambda^4 et rouge du soleil au coucher)

2. x $\sim$ 1 : Diffusion de Mie

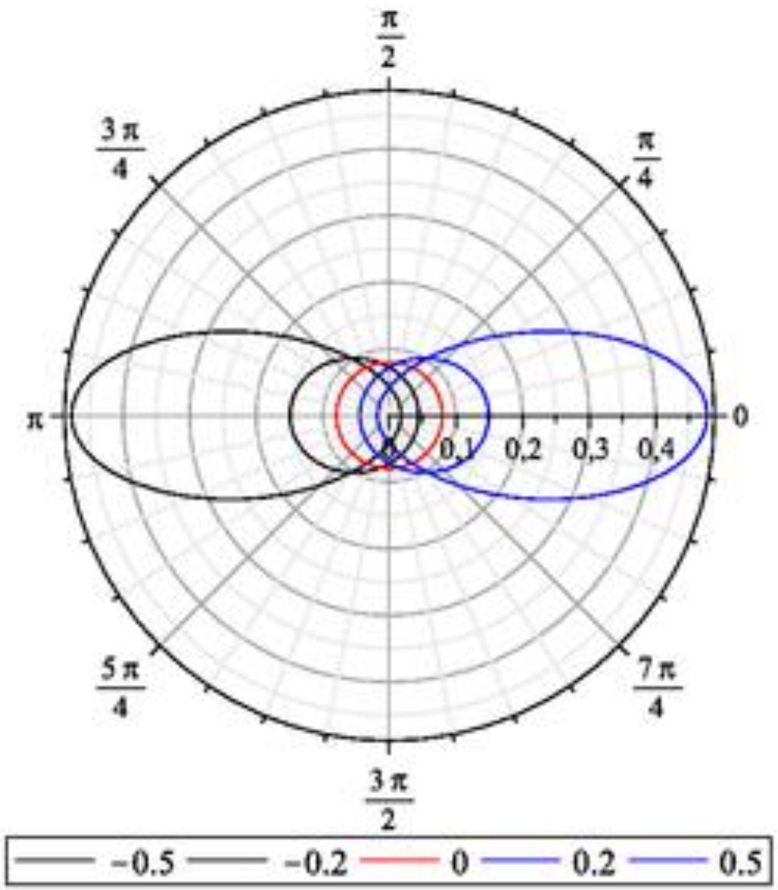
3. x>>1 : Diffusion géométrique (optique géométrique)

Or ici : x ≈ 10 donc diffusion de Mie caractérisée par g proche de 1

Fonction de phase de Henyey-Greenstein

La fonction de phase possède la symétrie azimutale : elle est donc fonction du seul angle de colatitude θ ou de son cosinus.

On prend g proche de 1



Equation LiDAR en puissance

Forme générale :

$P(z,\lambda)=K(\lambda)G(z)\eta(z, \lambda)T(z, \lambda)$

Avec $K(\lambda)=\frac{P_0(\lambda)cA_r\tau_T}{2}$ coefficient d'étalonnage

$G(z)=\frac{O(z)}{z^2}$ facteur de géométrie

$\eta(z, \lambda)$ coefficient de rétrodiffusion

$T(z, \lambda)=\exp(-2\int_0^z \alpha(z',\lambda)dz')$ transmission atmosphérique

Equation LiDAR en puissance

Prise en compte de l'énergie rétrodiffusée :

$E_{retro} = E_{LiDAR}$η(z)Δz avec η(z) : coefficient de rétrodiffusion (m^{-1})

Δz : la distance parcourue par le laser = $\frac{c\Delta t}{2}$

On divise par Δt :

$P_r = \frac{K}{z^2} e^{-2*0.02R^{0.6}z}$ avec $K = \frac{\rho c E_l A_r \tau_T \tau_R}{2}$ en Wm^{-2}

Code Python Déviation Gradient Thermique

```
K = 0.082
a = -2*10**9
b = 3*10**8
c = -2*10**7
d = 671830
e = -10751
h = 377.57
C = 1
Y0 = (15.3*10**-3,0)
T0 = 341.7

def f(Y,x):

(y,dy) = Y
T = a*y**5 + b*y**4 + c*y**3 + d*y**2 + e*y
+ h
dT = 5*a*y**4 + 4*b*y**3 + 3*c*y**2 + 2*d*y
+ e
return(dy, (-1/C**2)*(1+(K/T))*(K*dT/T**2))
```

```
def tracer_rayon(d,D,n):

""" d : longueur du gradient
D : distance gradient-écran"""
x = np.linspace(0,d,n)
y = si.odeint(f,Y0,x)
x2 = np.array([d+D])
(yf,dy_f) = y[n-1]
Y2 = np.array([dy_f*D + yf])
X = np.concatenate((x,x2))
Y = np.concatenate((y[:,0],Y2))
plt.plot(X,Y,"r",linewidth = 2)
plt.title("Trajectoire du rayon ")
plt.xlabel("x(m)")
plt.ylabel("y(m)")
plt.show()
```

Code Python Déviation Pluie (1/2)

```
import numpy as np
import matplotlib.pyplot as plt
import math
import random
from mpl_toolkits.mplot3d import Axes3D

#EN CM
#Ecran placé à ecran sur Ox

pos = np.array([0,0,0])
vect = np.array([1,0,0])

dist_moy = 30 # De 10 a 30
ecran = 1000
r = 0.3 #De 0.2 a 0.5
n_air = 1
n_eau = 1.33


def normalise(vect):
  return vect/np.linalg.norm(vect)

def deplace(pos,vect,d):
  return (pos+d*vect,vect)
```

```
def generer_goutte(pos,vect):
  """Retourne centre de la goutte + vecteur normal 
unitaire au pt d'intersect"""
  z = random.uniform(pos[2]-r , pos[2]+r)
  y = random.uniform(pos[1]-math.sqrt(r**2-(z-
pos[2])**2) , 
pos[1]+math.sqrt(r**2-(z-pos[2])**2))
  x = pos[0] + math.sqrt(r**2-(z-pos[2])**2-(y-
pos[1])**2)
  return (x,y,z),normalise(np.array([pos[0]-x,pos[1]-
y,pos[2]-z]))


def proba(n):
  for i in range(n):
     try:
        a,gouttes=pluie(pos,vect)
     except ValueError:
        print("erreur")
  plt.scatter(y,z,s=0.3,c="red")
  plt.scatter(0,0,s=0.3)
  axes = plt.gca()
  axes.set_xlim([-100,100])
  axes.set_ylim([-100,100])
  plt.plot()
```

Code Python Déviation Pluie (2/2)

```
def pluie(pos,vect):
  i=0
  while pos[0] < ecran:
    i+=1
    pos,vect = deplace(pos,vect,dist_moy)
    centre,normale=generer_goutte(pos,vect)
    pos,vect = refracte(pos,vect,normale,centre,n_air,n_eau)
  return pos,i


def refracte(point,incident,normale,centre,n1,n2):
  incident1 = normalise(incident)
  normale1 = normalise(normale)
  """En sortie: point et vecteur directeur du rayon lumineux refracté ou
réfléchi"""
  cos1 = -np.dot(normale1,incident1)
  rac_cos2 = 1-(n1/n2)**2*(1-cos1**2)
  if rac_cos2>=0:
    cos2 = math.sqrt(rac_cos2)
  else:
    """Reflexion totale"""
    return point,normalise((incident1 + (2*cos1)*normale1))

  refracte = (n1/n2)*incident1 + ((n1/n2)*cos1-cos2)*normale1
  refracte1 = normalise(refracte)
  pos2 = deplace(point,refracte1,2*r*(-
np.dot(normale1,refracte1)))[0]
```

```
"""ON RECOMMENCE POUR SORTIR DE LA GOUTTE"""
  normale2 = np.array([centre[0]-pos2[0],centre[1]-
pos2[1],centre[2]-pos2[2]])
  normale2 = normalise(normale2)

  cos1 = -np.dot(normale2,refracte1)
  rac_cos2 = 1-(n2/n1)**2*(1-cos1**2)
  if rac_cos2>=0:
    cos2 = math.sqrt(rac_cos2)
  else:
    """Reflexion totale"""
    raise ValueError('Reflexion totale interne!')
  if cos1>=0:
    refracte2 = (n2/n1)*refracte1 + ((n2/n1)*cos1-
cos2)*normale2
    refracte2 = normalise(refracte2)
    return pos2,refracte2
  else:
    refracte2 = (n2/n1)*refracte1 +
((n2/n1)*cos1+cos2)*normale2
    refracte2 = normalise(refracte2)
    return pos2,refracte2
```

Code Python Puissance (1/3)

```
def propagation1(N,L,Dx,Dy,g,Ma,Ms):
  W1=[] #énergie de chaque photon
  X1=[]
  Y1=[]
  I1=[]
  d1=[]
  for i in range(N):
    j=0
    w=h*c/l0 #énergie du photon au départ
    (x,y,z)=(0,0,0)
    (Ux,Uy,Uz)=(0,0,1)
    d=0
    while j<1000 and z<L and w>0:
      k=random.uniform(1,0)
      b=random.uniform(0,2*pi)
      a=HG(g,k)
      l=(-log(k))/(Ma+Ms)
      (Ux,Uy,Uz)=transfo(Ux,Uy,Uz,a,b)
      (x0,y0,z0)=(x,y,z)
      (x,y,z)=(x+l*Ux,y+l*Uy,z+l*Uz)
      w=w*(1-(Ms/(Ms+Ma)))
      d+=l
      j+=1
    (x,y)=positionecran(L,x0,y0,z0,x,y,z)
    if abs(x)<=Dx/2 and abs(y)<=Dy/2 and w>0 and z>=L: #le photon a atteint la cible
      W1.append(w)
      X1.append(x)
      Y1.append(y)
      I1.append([Ux,Uy,Uz])
      d1.append(d)
  n1=len(X1)
  return (W1,X1,Y1,I1,d1,n1)
```

Code Python Puissance (2/3)

```
RGB=["blue","green","orange","red","purple"]

def cadrage(m,M):
  if m!=M:
    return (1/(M-m),-m/(M-m))
  else:
    return(1,0)
```

```
def photonsecran2(W2,X2,Y2,Dx,Dy,N,d1,d2,t):
  W=[x for x in W2 if x!=0]
  print(W)
  (c,d)=cadrage(min(W),max(W))
  for i in range (len(W)):
        w0=int((c*W[i]+d)*(len(RGB)-1))
        #print(w0,W2[i])
        plt.scatter(X2,Y2,s=1,c=RGB[w0])
  plt.scatter(0,0,s=10,c="red")
  T=tempstot(d1,d2)
  P=puissance(N,t,T,W2)
  axes=plt.gca()
  axes.set_xlim([-Dx/2,Dx/2])
  axes.set_ylim([-Dy/2,Dy/2])
  plt.title(str(len(W))+" photons sur la
cible"+"("+str(N)+")"+"\n"+"Puissance relative reçue =
"+str(float(P)))
  plt.legend()
  plt.show()
```

Code Python Puissance (3/3)

```
def tempstot(d1,d2):
  D=[]
  for i in range (len(d1)):
    if d2[i]!=d1[i] and d2[i]!=0: #sinon le photon n'a pas
atteint le capteur
      D.append(d2[i])
  T=[x/c for x in D]
  return (T)


def puissance(N,t,T,W2):
  if T!=[]:
    (Tmax,Tmin)=(max(T),min(T))
    P1=(N*h*c)/(t*l0)  #pas de N au dénominateur ??
    P2=0
    if Tmax!=Tmin: #il y a plus de deux photons qui arrivent sur
le capteur
      E=sum(W2)
      P2=E/(Tmax-Tmin)
    return (P2/P1)
  return (None)
```

```
def puissancemoy(N,L,g,t,Dx,Dy,Ex,Ey,M,Ma,Ms):
  P=[]
  for i in range(M):
    (W1,X1,Y1,I1,d1,n1)=propagation1(N,L,Dx,Dy,g,Ma,Ms)
    (W2,X2,Y2,I2,d2,n2)=propagation2(L,Ex,Ey,g,W1,X1,Y1,I1,d1,n1,Ma,Ms)
    T=tempstot(d1,d2)
    p=puissance(N,t,T,W2)
    if p!=None and p>0 :
      P.append(p)
  if P!=[]:
    p0=sum(P)/len(P)
    return (p0)

def courbefinale(N,g,t,Ex,Ey,Dx,Dy,M):
  X=np.linspace(0.5,10,50)
  R=[5.00,12.5,25]
  Coeffs=[(0.0013,0.00132),(0.0024,0.00244),(0.0038,0.00387)]
  for j in range (len(R)):
    Y=[]
    for i in range (len(X)):
      p0=puissancemoy(N,X[i],g,t,Dx,Dy,Ex,Ey,M,Coeffs[j][0],Coeffs[j][1])
      if p0!=None:
        Y.append(p0)
      else:
        Y.append(0)
    plt.plot(X,Y,label=str(R[j])+"mm/h")
  plt.legend()
```

Bibliographie

[1] Savatier, F. : Vers la conduite automatique : Pour la Science, pp. 100-101. Janvier 2009

[2] du Chapelet, M., Fang, Z., Gauthier, Q., et al. : LiDAR : Etat de l'art, application en perception de l'environnement pour le véhicule autonome. : INSA Rouen. 2017

[3] Balland, B. : Optique géométrique: imagerie et instruments. : INSA Lyon, pp.89 - 92. 2007

[4] Concours d'admission Première épreuve de physique Filière MP. : Mines ParisTech 2012

[5] Flechon J. : Applications des franges non localisées - Vérification de la loi de Gladstone. : ENS Fontenay/St Cloud (1965) https://www.canalu.tv/video/cerimes/applications_des_franges_non_localisees_verification_de_la_loi_de_gladsto ne.12285

[6] Guo, J., Zhang, H. and Zhang, X. : Propagating Characteristics of Pulsed Laser in Rain. 2018

[7] Ian Wong, J. : Driverless cars have a new way to navigate in rain or snow. : Quartz.com 2016

Références images

D1: https://phys.org/news/2017-12-autonomous-vehicles-uncertain-effects.html

D4:https://www.powerbulbs.com/eu/blog/2015/09/tips-on-driving-in-fog

D5:https://fr.wikipedia.org/wiki/Lidar#Lidars_t%C3%A9l%C3%A9m%C3%A8tres_laser_%C3%A0_balayage

D6 : http://slideplayer.fr/slide/1579892/

D10: https://www.flickr.com/photos/lawatt/47803095

D14: https://fr.wikipedia.org/wiki/Th%C3%A9orie_de_Mie

D23: https://dailygeekshow.com/route-chaleur-hiver-ete/?utm_medium=social&utm_source=pinterest

D23: https://lawrencestevens.com/blog/2014/7/31/road-trip-day-7-el-paso

D26: https://www.numerama.com/business/166851-voiture-autonome-google-uber-volvo-ford-et-lyft-font-alliance.html

D27: http://lemobiliste.com/voiture-autonome-attente-mondiale/

D28: https://www.fudosansell-agent.com/useful/document/

D34: https://fr.m.wikipedia.org/wiki/Fonction_de_phase_de_Henyey-Greenstein